Lossless Simplification of Object-Centric Declarative Process Models

Aaron Küsters and Wil M.P. van der Aalst

Chair of Process and Data Science (PADS), RWTH Aachen University
{kuesters,wvdaalst}@pads.rwth-aachen.de

Abstract. Declarative object-centric process modeling formalisms like OC-Declare capture synchronization patterns across object types that per-type formalisms cannot express. However, this expressiveness comes at a cost: Discovered models are often large and structurally complex. In this paper, we address this problem for OC-Declare *existence constraint* models, i.e., the sub-family of constraints requiring that a related target event exists, by presenting a *lossless transitive reduction* that removes constraints implied by others through transitivity, accounting for the structural features specific to OC-Declare such as arrow-type domination and object involvement annotations. We prove that this reduction preserves language equivalence, i.e., does not compromise the fitness or precision of models. Building on this reduction, we present a lossless activity projection that enables focusing on a subset of activities while preserving all transitive dependencies among them. An evaluation on three real-life and three simulated datasets shows that the lossless reduction is highly effective in practice: It reduces the size of models by over 60% on average for complete discovery and produces models with structural complexity comparable to per-type formalisms without sacrificing expressiveness or precision. Together, these techniques, supported by an open-source implementation, allow combining the completeness guarantees of OC-Declare discovery with structurally simple models, paving the way for practical process analysis using OC-Declare.

Keywords: Object-centric process mining · Declarative process models

1 Introduction

Process mining aims to extract insights from event data recorded in information systems during the execution of business processes. While event data was traditionally analyzed case-by-case, Object-Centric Process Mining (OCPM) analyzes the data of interacting entities together, better capturing real-life processes where multiple objects of different types interact.

Multiple object-centric process modeling approaches have been proposed, including object-centric Directly-Follows-Graphs (DFGs) [17], Petri nets [1, 10], OCBC [2], DCR graphs [6], and OC-Declare [14]. Most procedural object-centric formalisms (OC-DFGs, OC-Petri nets) discover separate models per object type and combine them. They cannot directly express how different object

Fig. 1. An OC-DECLARE model for an order management process before and after lossless transitive reduction. Each arc represents a constraint between two activities, annotated with how objects of different types must be shared. The three crossed-out constraints are implied by the remaining arcs through transitivity and are removed without changing the model's semantics, yielding the simpler model on the bottom.

types synchronize within and across activities. OC-DECLARE is a declarative formalism that captures exactly such cross-type synchronization constraints. Figure 1 shows two example models: Each arc represents a constraint between two activities, annotated with how objects of different types must be shared. For instance, the arc from `place order` to `confirm order` requires that for each `place order` event involving an order, there is a later `confirm order` event involving the same order, and all associated items: A constraint spanning multiple object types that per-type models cannot express.

However, this expressiveness comes at a cost: Discovered OC-DECLARE models are often large and structurally complex, with many redundant arcs, as the additional dimensions of multiple *object types* and their *interactions* produce many constraints. Simpler models are valuable both for human understanding [11] and for computational efficiency in downstream tasks such as conformance checking, including automated and AI-assisted analysis, where model size limits what can be processed at once. For case-centric formalisms, reduction techniques such as implicit-place removal in Petri nets [9], hierarchy-based Declare reduction [15], and transitive reduction for declarative models [7] have been proposed. However, so far, there are *no dedicated reduction techniques for object-centric process models*. We address this gap by presenting techniques for simplifying OC-DECLARE existence constraint models. We focus on existence constraints, the sub-family of OC-DECLARE that requires the existence of related target events and the only family for which automatic discovery currently exists. Our lossless transitive reduction removes constraints implied by combinations of other constraints, analogous to how implicit places are removed from Petri nets. The bottom of Figure 1 shows the result: Fewer constraints but identical semantics. Our contributions are: (1) A *lossless transitive reduction* for OC-DECLARE existence constraint models that accounts for arrow type and object involvement domina-

tion, with proven language equivalence. (2) A lossless *activity projection* built on the reduction that preserves transitive dependencies when focusing on a subset of activities. (3) An evaluation on six datasets quantifying the effectiveness of lossless reduction, the constraint loss of naïve vs. complete activity projection, and the structural complexity of reduced models relative to OC-DFGs and OC-Petri nets, together with an open-source implementation of all techniques. The remainder covers related work (Sec. 2), preliminaries (Sec. 3), the reduction techniques (Sec. 4), the evaluation (Sec. 5), and the conclusion (Sec. 6).

2 Related Work

Object-Centric Process Models. Procedural object-centric formalisms include directly-follows graphs [17] and Petri nets [1, 10]; declarative ones include OCBC [2], OC-DCR graphs [6], and OC-DECLARE [14]. A shared challenge is the complexity of models. The object dimension typically leads to large constraint/arc counts, yet only a few approaches specify the complexity of discovered models across datasets. To our knowledge, no formal simplification technique for object-centric process models has been proposed to date.

Simplification of Declarative Process Models. For case-centric DECLARE, Maggi et al. [15] apply hierarchy-based reduction, dropping weaker constraints subsumed by stronger templates. Semantical vacuity detection can identify constraints whose activation trigger does not occur in traces [16], but does not directly transfer to the object-centric setting and the event-based semantics of OC-DECLARE. Di Ciccio and Mecella [8] use transitive reduction during DECLARE discovery, and Di Ciccio et al. [7] resolve redundancies on a simple directed constraint graph. Our work extends transitive reduction to the object-centric setting, accounting for arrow type and object involvement domination (ALL/EACH/ANY). Integrating inconsistency-detection is left for future work. For imperative models, lossless simplification such as implicit place removal [9] is well-established but not yet extended to the object-centric setting.

Complexity Metrics. Andaloussi et al. [4] define graph-based complexity metrics for DCR graphs and show through a user study that higher structural complexity correlates with lower understandability. Although these metrics do not account for the object dimension, we adopt them to quantify the basic structural complexity of OC-DECLARE models in Sec. 3.

3 Preliminaries

We assume pairwise disjoint universes of events $\mathcal{E}$, objects $\mathcal{O}$, event types (activities) $\mathcal{ET}$, object types $\mathcal{OT}$, and timestamps $\mathbb{T}$. Following a subset of OCEL 2.0 [5], an Object-Centric Event Log (OCEL) is a tuple $L = (E, O, R, \mathit{type}, \mathit{time})$ with events $E \subseteq \mathcal{E}$, objects $O \subseteq \mathcal{O}$, E2O relations $R \subseteq E \times O$, a type assignment $\mathit{type} \in E \cup O \to \mathcal{ET} \cup \mathcal{OT}$ ($\mathit{type}(e) \in \mathcal{ET}$ for $e \in E$, $\mathit{type}(o) \in \mathcal{OT}$ for $o \in O$), and a timestamp function $\mathit{time} \in E \to \mathbb{T}$. Table 1 shows an OCEL of an order management process involving objects of types `orders`, `items`, and `employees`.

Table 1. An example OCEL of an order process with object types `orders` (ord_1), `items` (it_1, it_2), and `employees` (emp_1, emp_2).

Event	Activity	Timestamp	E2O Objects
e_1	place order	01.01.25 10:00	$\{ord_1, it_1, it_2, emp_1, emp_2\}$
e_2	confirm order	01.01.25 13:00	$\{ord_1, it_1, it_2, emp_2\}$
e_3	pick item	01.01.25 16:00	$\{it_1, emp_1\}$
e_4	create package	02.01.25 11:00	$\{it_1, it_2, emp_1, emp_2\}$

For an OCEL $L = (E, O, R, type, time)$, we use the following shorthands: $E_L = E$, $O_L = O$; $e <_L e'$ iff $time(e) < time(e')$; $E_L^{et} = \{e \in E \mid type(e) = et\}$ and $O_L^{ot} = \{o \in O \mid type(o) = ot\}$ for the events of type et and objects of type ot; $obj_L(e) = \{o \in O \mid (e, o) \in R\}$ for the objects related to e, and $obj_L^{ot}(e) = obj_L(e) \cap O_L^{ot}$ for those of type ot.

3.1 OC-Declare Existence Constraints

OC-Declare models consist of a set of arcs [14]. We focus on *existence constraints* (imposing existence of events; minimal/maximal counts $1/\infty$), the family for which automatic discovery exists [14]. We consider arrow types ●—, ●→, ●←, excluding directly-follows arcs, which do not allow for transitive reduction, and object-to-object relations. Negative constraints (forbidding target events) are left for future work, also as no discovery approach exists for them yet.

Definition 1. *An existence OC-Declare arc is a tuple $d = (ar, s, t, oi)$ of:*
- *An arrow type $ar \in \{●—, ●→, ●←\}$, specifying a temporal restriction: ●— (*association*) imposes no temporal order, ●→ (*eventually-follows*) requires the target event to occur after the source event, and ●← (*eventually-precedes*) requires it to occur before.*
- *A source activity $s \in \mathcal{ET}$, specifying the activity triggering the constraint.*
- *A target activity $t \in \mathcal{ET}$, specifying the activity of the target events.*
- *A partial object involvement function $oi \in \mathcal{OT} \not\to \{\textsc{Each}, \textsc{All}, \textsc{Any}\}$, specifying how objects of the type must be shared between source and target events: All requires all objects of that type to be present, Each requires existence for each object individually, and Any requires at least one shared object.*

For convenience, we also write $ar_d = ar$, $oi_d = oi$, as well as $oi^X = \{ot \in \text{dom}(oi) \mid oi(ot) = X\}$ for $X \in \{\textsc{Each}, \textsc{All}, \textsc{Any}\}$.

The top part of Figure 1 shows an OC-Declare model for the order process from Sec. 1. As an example arc, consider d_1 from `place order` to `confirm order` with $ar = ●→$, $oi^{\textsc{Each}} = \{\texttt{orders}\}$, $oi^{\textsc{All}} = \{\texttt{items}\}$: For each `place order` event and each involved order, there must be a later `confirm order` event involving that order and all items of the source event. In Table 1, e_1 satisfies d_1 via e_2, which involves ord_1 and all items of e_1. The crossed-out constraints on the top of Figure 1 are, as we show later, implied by combining other constraints. Removing them leads to the simpler but equivalent model at the bottom.

Next, we define the semantics of OC-DECLARE existence constraints. The definition is simplified from [14] for the chosen scope and slightly adapted to handle empty object sets consistently across EACH, ALL, and ANY.

Definition 2. *Let $d = (ar, s, t, oi)$ be an OC-DECLARE existence constraint. In the context of an OCEL L, we define when an event $e \in E_L^s$ (i.e., of the source activity s) satisfies d (written as $e \models_L d$). We write $ot_1, \ldots, ot_n$ for all non-empty EACH-involved object types (i.e., $\{ot \in oi^{\text{EACH}} \mid obj_L^{ot}(e) \neq \emptyset\} = \{ot_1, \ldots, ot_n\}$).*

$$e \models_L d \Leftrightarrow \forall_{o_1 \in obj_L^{ot_1}(e), \ldots, o_n \in obj_L^{ot_n}(e)} E_L^t \cap F_{\text{TIME}} \cap F_{\text{EACH}} \cap F_{\text{ALL}} \cap F_{\text{ANY}} \neq \emptyset$$

$$\text{where } F_{\text{TIME}} = \begin{cases} E_L, & \text{if } ar = \bullet\!\!-\!\!- \\ \{e' \in E_L \mid e' >_L e\}, & \text{if } ar = \bullet\!\!-\!\!\rightarrow \\ \{e' \in E_L \mid e' <_L e\}, & \text{if } ar = \bullet\!\!\leftarrow\!\!- \end{cases}$$

$$F_{\text{EACH}} = \{e' \in E_L \mid \{o_1, \ldots, o_n\} \subseteq obj_L(e')\}$$

$$F_{\text{ALL}} = \{e' \in E_L \mid \forall_{ot \in oi^{\text{ALL}}}\ obj_L^{ot}(e) \subseteq obj_L^{ot}(e')\}$$

$$F_{\text{ANY}} = \{e' \in E_L \mid \forall_{ot \in oi^{\text{ANY}}}\ obj_L^{ot}(e) \neq \emptyset \Rightarrow obj_L^{ot}(e) \cap obj_L^{ot}(e') \neq \emptyset\}$$

For events of different types $e \in E_L \setminus E_L^s$, we define $e \models_L d$ to hold trivially.

Put simply, an event e satisfies a constraint d if for every combination of its EACH-involved objects, there exists a target event e' that: (1) satisfies the temporal rule, (2) involves all of these EACH objects, (3) involves all objects required by ALL, and (4) involves at least one object from every set required by ANY.

An OC-DECLARE model $(A, \mathcal{D})$ is a set $\mathcal{D}$ of OC-DECLARE constraints over a set of activities $A \subseteq \mathcal{ET}$.

Definition 3 (Model Language). *Let $(A, \mathcal{D})$ be an OC-DECLARE model and let L be an OCEL over events from A (i.e., $\forall_{e \in E_L}\ type_L(e) \in A$). For an $e \in E_L$, we write $e \models_L \mathcal{D} \Leftrightarrow \forall_{d \in \mathcal{D}}\ e \models_L d$ and $L \models \mathcal{D} \Leftrightarrow \forall_{e \in E_L}\ e \models_L \mathcal{D}$, when L fulfills the model $(A, \mathcal{D})$. For a single constraint d, we write $L \models d$ as shorthand for $L \models \{d\}$. The language of $(A, \mathcal{D})$ is then defined as the set of all OCELs that satisfy these criteria: $\mathcal{L}((A, \mathcal{D})) = \{L \mid \forall_{e \in E_L}\ (e \models_L \mathcal{D} \wedge type_L(e) \in A)\}$.*

An Apriori-style discovery approach for OC-DECLARE was presented in [14], discovering the strictest constraints between each activity pair that are satisfied by at least a fraction of $1 - \tau$ events for noise threshold $\tau \in [0, 1]$.

3.2 Complexity Metrics

We adapt the graph-based metrics from [4], treating an OC-DECLARE model $G = (A, \mathcal{D})$ as a directed graph (activities as nodes, constraints as edges). Let $Comp(G)$ be the set of weakly connected components and $A_C, \mathcal{D}_C$ the activities/constraints of a component C. With $T = \{\bullet\!\!-\!\!-, \bullet\!\!-\!\!\rightarrow, \bullet\!\!\leftarrow\!\!-\}$ and $p(C, r) = |\{d \in \mathcal{D}_C \mid ar_d = r\}|/|\mathcal{D}_C|$, the metrics are: *size* $S(G) = |A| + |\mathcal{D}|$, *density* $D(G) = \max_{C \in Comp(G)} |\mathcal{D}_C|/|A_C|$ (constraints per activity in the densest component), *separability* $Sep(G) = |Comp(G)|/S(G)$ (share of independent components), and *constraint variability* $CV(G) = \max_{C \in Comp(G) \wedge |\mathcal{D}_C| > 0} \{$

$-\sum_{r \in T} p(C,r) \log_{|T|} p(C,r)\big\}$ (entropy over arrow types). Higher values indicate higher complexity (except separability). These graph-structural metrics do not capture object-centric aspects. Thus, we complement them with *typed connections* $TC(G) = \sum_{d \in \mathcal{D}} |\text{dom}(oi_d)|$, summing object type annotations across all arcs. In OCPNs [1] and OC-DFGs [17], TC coincides with the arc count, as each arc is annotated with exactly one object type.

4 Simplification of OC-Declare Models

In this section, we present a lossless transitive reduction for OC-Declare that removes redundant constraints while preserving the model semantics. Building on this, we additionally define a lossless activity projection that preserves transitive dependencies when focusing on a subset of activities.

4.1 Lossless Transitive Reduction

A standard graph transitive reduction does not directly apply to OC-Declare, since constraint arcs carry arrow types and object involvement annotations. We define two corresponding domination relations. An object involvement is dominated by another if every object type carries at least the same strictness, ordered All $\succ$ Each $\succ$ Any: All (event involves all objects of the type) implies Each (one event for each object), which in turn implies Any (some object shared).

Definition 4 (Object Involvement Domination). *Consider two object involvements $oi, oi' \in \mathcal{OT} \not\to \{\textsc{Each}, \textsc{All}, \textsc{Any}\}$. Then oi dominates oi' (written as $oi' \preceq oi$) if ${oi'}^{\textsc{All}} \subseteq oi^{\textsc{All}}$, ${oi'}^{\textsc{Each}} \subseteq oi^{\textsc{All}} \cup oi^{\textsc{Each}}$, and ${oi'}^{\textsc{Any}} \subseteq oi^{\textsc{All}} \cup oi^{\textsc{Each}} \cup oi^{\textsc{Any}}$.*

There is a similar domination relation for the temporal constraints imposed by arrow types. For example, a ●→ constraint is stronger than the same constraint with ●—, as it requires the target event to occur after the source event.

Definition 5 (Arrow Domination). *Let $ar, ar' \in \{\bullet\!\!-\!\!-, \bullet\!\!-\!\!\!\rightarrow, \bullet\!\!\leftarrow\!\!-\}$. We write $ar' \preceq ar$ if $ar = ar'$ or $ar' = \bullet\!\!-\!\!-$.*

If a constraint d' is dominated by another constraint d in terms of object involvement and arrow type, d' is implied by d [14]. By design, the Apriori-style discovery approach only discovers the strictest object involvements per arc.

Lemma 1 (Domination Implication). *Let $d_1 = (ar_1, a, c, oi_1)$ be an OC-Declare constraint and let $d' = (ar', a, c, oi')$ with $ar' \preceq ar_1$ and $oi' \preceq oi_1$. Then for any OCEL L it holds that $L \models d_1 \Rightarrow L \models d'$.*

Proof. Fix OCEL $L \models d_1$ and event $e_1 \in E_L^a$. Write $ot_1, \ldots, ot_m$ for the types in ${oi'}^{\textsc{Each}}$ with $obj_L^{ot_i}(e_1) \neq \emptyset$ (empty Each types impose no requirement per Definition 2) and fix $o_i \in obj_L^{ot_i}(e_1)$ for each i. As $oi' \preceq oi_1$, each ot_i lies in $oi_1^{\textsc{All}} \cup oi_1^{\textsc{Each}}$, and since $obj_L^{ot_i}(e_1) \neq \emptyset$, each $ot_i \in oi_1^{\textsc{Each}}$ is among the non-empty Each types quantified over for d_1. Choose a combination for d_1 that

assigns o_i to every such ot_i. Since $L \models d_1$, there exists $e_2 \in E_L^c$ satisfying d_1 for e_1 and this combination, which gives $\{o_1, \ldots, o_m\} \subseteq obj_L(e_2)$ in either case for ot_i: for $ot_i \in oi_1^{\text{EACH}}$, $e_2 \in F_{\text{EACH}}$ contains the assigned o_i; for $ot_i \in oi_1^{\text{ALL}}$, $e_2 \in F_{\text{ALL}}$ gives $obj_L^{ot_i}(e_1) \subseteq obj_L^{ot_i}(e_2)$. We verify that e_2 satisfies all filters of d' for e_1. **Temporal:** $ar' \preceq ar_1$ gives $ar' =$ ●— (no filter) or $ar' = ar_1$ (inherited from d_1). **Activity:** $e_2 \in E_L^c$ by construction. **Object involvement:** For $ot \in \text{dom}(oi')$, by $oi' \preceq oi_1$: if $ot \in {oi'}^{\text{ALL}}$ then $ot \in oi_1^{\text{ALL}}$, hence $obj_L^{ot}(e_1) \subseteq obj_L^{ot}(e_2)$; if $ot = ot_i \in \{ot_1, \ldots, ot_m\}$ then $o_i \in obj_L^{ot}(e_2)$ by construction; if $ot \in {oi'}^{\text{ANY}}$, either $obj_L^{ot}(e_1) = \emptyset$ (F_{ANY} of d' holds vacuously), or $oi' \preceq oi_1$ places ot in oi_1 as ALL, EACH, or ANY and e_1, e_2 share a type-ot object: F_{ALL} gives $obj_L^{ot}(e_1) \subseteq obj_L^{ot}(e_2)$, F_{EACH} puts the quantified object of $obj_L^{ot}(e_1)$ into e_2, and F_{ANY} of d_1 gives a shared object. Thus $e_1 \models_L d'$. □

However, there are also reduction opportunities that go beyond such a direct implication. For example, consider the arc from `place order` to `pay order` in Figure 1. Intuitively, this arc can be removed using a transitive reduction based on the `confirm order` step in between. We consider this reduction for paths of arbitrary length. Figure 2 visualizes the general case for paths of length $n \geq 1$. For $n = 1$, this corresponds to direct dominance of two arcs (as in Lemma 1).

Definition 6 (Transitivity Relation). *Let $n \geq 1$ and let $a, b_1, \ldots, b_{n-1}, c \in \mathcal{ET}$. Let $d_1, \ldots, d_n$ be a path of OC-DECLARE arcs with $d_i = (ar_i, b_{i-1}, b_i, oi_i)$, where we write $b_0 = a$ and $b_n = c$. Moreover, consider another OC-DECLARE constraint $d_{ac} = (ar_{ac}, a, c, oi_{ac})$ with (1) $\forall_{1 \leq i \leq n}\ d_{ac} \neq d_i$, (2) $\forall_{1 \leq i \leq n}\ ar_{ac} \preceq ar_i$, (3) $\forall_{1 \leq i \leq n}\ oi_{ac} \preceq oi_i$, and (4) $\forall_{2 \leq i \leq n}\ oi_{ac}^{\text{ANY}} \cap oi_i^{\text{ANY}} = \emptyset$. Then d_{ac} is in the transitive relation of $d_1, \ldots, d_n$ and we write $d_{ac} \in Trans(\{d_1, \ldots, d_n\})$.*

For example, in Figure 1, the three crossed-out constraints are in the transitive relation of others through intermediate activities like `confirm order`.

Definition 7 (Reduction). *Let $\mathcal{D}$ be a set of OC-DECLARE arcs. A transitive reduction $\mathcal{D}\downarrow$ of $\mathcal{D}$ is a subset-minimal set $\mathcal{D}\downarrow \subseteq \mathcal{D}$ such that every removed constraint is implied by retained ones: $\forall_{d \in \mathcal{D} \setminus \mathcal{D}\downarrow} \exists_{d_1, \ldots, d_n \in \mathcal{D}\downarrow}\ d \in Trans(\{d_1, \ldots, d_n\})$.*

This definition generalizes the well-known transitive reduction over directed acyclic graphs [3] to the annotated constraint graphs of OC-DECLARE. While the reduction is unique for acyclic graphs, for general OC-DECLARE models the reduction order may influence the resulting reduced model [3]. In the remainder, we consider $\mathcal{D}\downarrow$ to be the (fixed, unique) transitive reduction obtained by

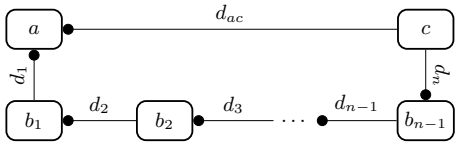

Fig. 2. An OC-DECLARE model shown as a graph, with two paths from activity a to c: Direct, over constraint arc d_{ac} and indirect over n constraints $d_1, \ldots, d_n$.

iteratively removing constraints that are in the transitive relation of still-active others, in deterministic order, until none remain. Next, we show that all such reductions are language equivalent to the original model. First, the reduction allows at least the same behavior as the original.

Lemma 2 (Fitness Preservation). *Let $(A, \mathcal{D}\downarrow)$ be a transitive reduction of an OC-DECLARE model $(A, \mathcal{D})$. Then for any OCEL L: $L \models \mathcal{D} \Rightarrow L \models \mathcal{D}\downarrow$ and thus $\mathcal{L}((\mathcal{A}, \mathcal{D})) \subseteq \mathcal{L}((\mathcal{A}, \mathcal{D}\downarrow))$.*

This holds trivially, as $\mathcal{D}\downarrow \subseteq \mathcal{D}$ and constraints are only removed, not added (Definition 3). More importantly, removed constraints are already implied by others in the original model, so the reduction removes no behavioral restriction. Using Lemma 1, we now prove the general reduction implication for paths of arbitrary length. The key insight specific to the object-centric setting is that ANY object involvements are *not* transitive: For three events e_1, e_2, e_3, $obj_L^{ot}(e_1) \cap obj_L^{ot}(e_2) \neq \emptyset$ and $obj_L^{ot}(e_2) \cap obj_L^{ot}(e_3) \neq \emptyset$ does not imply $obj_L^{ot}(e_1) \cap obj_L^{ot}(e_3) \neq \emptyset$. This is precisely why condition (4) in Definition 6 requires $oi_{ac}^{\text{ANY}} \cap oi_i^{\text{ANY}} = \emptyset$ for intermediate constraints $i \geq 2$, ensuring ANY types are "caught" by ALL or EACH en route.

Lemma 3 (Reduction Implication). *Consider an OC-DECLARE constraint path $d_1, \ldots, d_n$ and a constraint $d' \in \mathit{Trans}(\{d_1, \ldots, d_n\})$. Then for any OCEL L it holds that $L \models \{d_1, \ldots, d_n\} \Rightarrow L \models d'$.*

Proof. By induction on n. *Base case ($n = 1$)*: follows from Lemma 1, as conditions (2),(3) of Definition 6 give $ar' \preceq ar_1$ and $oi' \preceq oi_1$. *Base case ($n = 2$)*: Let $d_1 = (ar_1, a, b, oi_1)$, $d_2 = (ar_2, b, c, oi_2)$, $d' = (ar', a, c, oi')$. Fix $e_1 \in E_L^a$ and, as in Lemma 1, $o_i \in obj_L^{ot_i}(e_1)$ for the types $ot_1, \ldots, ot_m \in oi'^{\text{EACH}}$ with $obj_L^{ot_i}(e_1) \neq \emptyset$. Since $oi' \preceq oi_1$, each $ot_i \in oi_1^{\text{ALL}} \cup oi_1^{\text{EACH}}$; as in Lemma 1, fix $e_2 \in E_L^b$ satisfying d_1 for e_1 with $\{o_1, \ldots, o_m\} \subseteq obj_L(e_2)$. For each $ot \in oi'^{\text{ANY}}$ with $obj_L^{ot}(e_1) \neq \emptyset$ (empty ones satisfy F_{ANY} of d' vacuously), $oi' \preceq oi_1$ places ot in oi_1 as ALL, EACH, or ANY; as in Lemma 1, each case yields an object of type ot shared by e_1 and e_2, and we fix one, $o_{ot} \in obj_L^{ot}(e_1) \cap obj_L^{ot}(e_2)$. The objects $o_1, \ldots, o_m$ and all fixed o_{ot} are involved in e_2, and their types lie in $oi_2^{\text{ALL}} \cup oi_2^{\text{EACH}}$: immediate from $oi' \preceq oi_2$ for $ot_1, \ldots, ot_m$, while for the ANY types $oi' \preceq oi_2$ also permits oi_2^{ANY}, which condition (4) of Definition 6 excludes. As in Lemma 1, fix a single $e_3 \in E_L^c$ satisfying d_2 for e_2 involving all these objects. Then e_3 satisfies d' for e_1: the EACH filter by $\{o_1, \ldots, o_m\} \subseteq obj_L(e_3)$; the ANY filter by $o_{ot} \in obj_L^{ot}(e_1) \cap obj_L^{ot}(e_3)$; the ALL filter as $oi'^{\text{ALL}} \subseteq oi_1^{\text{ALL}} \cap oi_2^{\text{ALL}}$ gives $obj_L^{ot}(e_1) \subseteq obj_L^{ot}(e_2) \subseteq obj_L^{ot}(e_3)$; the temporal filter by $ar' \preceq ar_1, ar_2$ and transitivity of $<_L$. Hence $e_1 \models_L d'$. *Induction step ($n \to n + 1$)*: Construct $d'' = (ar', b_0, b_n, oi')$. Conditions (1)–(4) transfer from d' to d'' for the sub-path $d_1, \ldots, d_n$, so $d'' \in \mathit{Trans}(\{d_1, \ldots, d_n\})$ (if $d'' = d_i$ for some i, $L \models d''$ holds directly). By induction, $L \models \{d_1, \ldots, d_n\} \Rightarrow L \models d''$. The pair d'', d_{n+1} satisfies the $n = 2$ case (with $oi'^{\text{ANY}} \cap oi_{n+1}^{\text{ANY}} = \emptyset$ by condition (4)), yielding $L \models d'$. □

As all constraints removed in the reduction are implied by others, the precision does not change. Combined with Lemma 2 this yields language equivalence.

Theorem 1 (Language Equivalence). *Let $(A, \mathcal{D})$ be an OC-DECLARE model and $\mathcal{D}\downarrow$ a transitive reduction of $\mathcal{D}$. Then $\mathcal{L}((\mathcal{A}, \mathcal{D}\downarrow)) = \mathcal{L}((\mathcal{A}, \mathcal{D}))$.*

The forward direction ($\supseteq$) follows from Lemma 2. For ($\subseteq$), if $L \models \mathcal{D}\downarrow$, then Definition 7 and Lemma 3 imply $L \models d$ for every $d \in \mathcal{D} \setminus \mathcal{D}\downarrow$, thus $L \models \mathcal{D}$.

In the implementation, arcs are processed in a fixed sorted order (deterministically). An arc is dropped if another path of still-active arcs from its source to its target dominates it (condition (4) from the second hop on). Each of the m candidates triggers a BFS that scans at most m arcs with an $O(r)$ domination check per edge, giving $O(m^2 r)$ time ($m = |\mathcal{D}|$, $r = |\mathcal{OT}|$). The reduction completes in under 0.3 s on all evaluated datasets.

4.2 Activity Projection

Building on the transitive reduction, we can now define a lossless activity projection. Focusing analysis on a subset $A' \subseteq A$ of activities by naïvely dropping constraints touching excluded activities can lose transitive dependencies; e.g., projecting Figure 1 onto $A' = \{\texttt{place order}, \texttt{confirm order}, \texttt{create package}\}$ disconnects `create package` (Figure 3). We define an *activity projection* that reuses the transitivity mechanism from Definition 6 to recover such implied arcs before filtering, preserving all transitive dependencies between activities in A'.

Definition 8 (Activity Projection). *Let $(A, \mathcal{D})$ be an OC-DECLARE model and $A' \subseteq A$ a subset of activities. The* complete activity projection *$(A', \mathcal{D}')$ is constructed as follows: 1) Build the* transitive closure *of $\mathcal{D}$ as $\mathcal{D}^+ = \mathcal{D} \cup \{d \in Trans(\{d_1, \ldots, d_n\}) \mid \{d_1, \ldots, d_n\} \subseteq \mathcal{D}\}$, i.e., the superset of $\mathcal{D}$ in which all transitive dependencies are explicitly included. 2) Filter the constraints to only those between activities in A': $\mathcal{F} = \{(ar, s, t, oi) \in \mathcal{D}^+ \mid s, t \in A'\}$, and 3) Perform the transitive reduction $\mathcal{D}' = \mathcal{F}\downarrow$.*

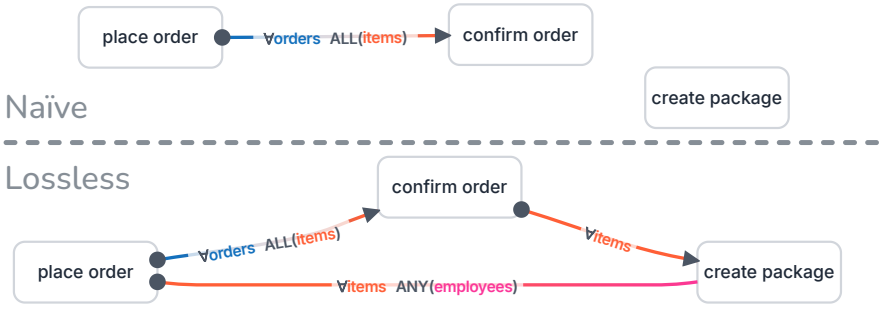

Fig. 3. Activity projection of the reduced model from Figure 1 onto $A' = \{\texttt{place order}, \texttt{confirm order}, \texttt{create package}\}$. In the naïve projection `create package` is disconnected, as there is no direct connection to it. The lossless projection recovers transitive dependencies through the excluded intermediary activities.

5 Evaluation

We implemented the reduction and evaluation in open-source code (`https://github.com/aarkue/oc-declare-red/`). The implementation is also integrated in Rust4PM [13] and available from Python, Rust, and graphical tools.

We evaluate our approach on six datasets, summarized in Table 2. The datasets comprise three real-life BPIC logs (converted via [12]) and three simulated logs (from `https://ocel-standard.org/`). We adapted the OC-DECLARE discovery from [14] with noise threshold $\tau = 0.2$ in three modes: Complete (CO), which applies no filtering, and two modes applying the lossy bounded-matching filter of [14], which drops constraints whose matching set of target events exceeds a threshold (here $N = 20$) and whose object involvement is thus too weak to be informative, either during discovery (pre-pruning, PR) or afterwards (post-pruning, PO). The evaluated arcs are existence constraints per Sec. 3 where discovery is available. Negative and directly-follows constraints are out of scope.

Lossless Reduction. Figure 4 reports the metric improvements per discovery mode. For CO, i.e., without any lossy filtering, constraint count, model size, and density improve by >60% on average and up to >80% on some datasets. Typed connections decrease slightly less, since surviving constraints carry more object types. Variability improves by about 10%, while separability is essentially unchanged. The number of constraints and typed connections after the reduction are shown in Table 2. The reduction took <0.3 s in all runs.

Fig. 4. Improvements in complexity measures after lossless reduction, aggregated over datasets and shown per discovery mode (each model $\mathcal{D}$ vs. its lossless reduction $\mathcal{D}\downarrow$).

Cross-Formalism Comparison. Table 2 compares *typed connections* across formalisms (sum of object involvements for OC-DECLARE; arc counts for OC-DFGs/OCPNs). After reduction, OC-DECLARE has the fewest typed connections on most logs, even though it captures inter-object synchronization the other formalisms cannot express. For OC-DCR graphs [6] discovery could not be run systematically. The original work reports 440 constraints on BPIC2017.

Table 2. Dataset properties and constraint/typed connection (TC) counts after reduction $\mathcal{D}\downarrow$ for modes CO, PO, PR. Last two columns: OC-DFG/OCPN [1, 17] via pm4py.

Dataset	$\lvert\mathcal{ET}\rvert$	$\lvert\mathcal{OT}\rvert$	$\lvert E\rvert$	$\lvert O\rvert$	$\lvert\mathcal{D}\downarrow\rvert$			Typed Connections $TC(\mathcal{D}\downarrow)$				
					CO	PO	PR	CO	PO	PR	*OC-DFG*	*OCPN*
P2P	10	7	14671	9543	19	19	19	30	30	30	52	86
O2C	11	6	21008	10840	70	55	25	152	128	37	244	160
Logistics	14	7	35413	13910	42	35	34	59	52	41	76	134
BPIC2014	330	7	690622	228885	1974	2422	2573	3050	3454	3165	16836	3394
BPIC2017	26	4	1202267	106162	110	48	49	121	59	49	412	150
BPIC2019	42	4	1595923	330685	276	199	243	622	509	388	2708	362

Activity Projection. When the reduced CO models are projected onto activity sets of size $k = 2, \ldots, 8$ ($n = 10$ random samples per k, or top-k by frequency), the naïve approach misses 48%–74% of constraints on random subsets and 10%–36% on top-k, whereas the lossless projection recovers all transitive dependencies through excluded intermediaries.

Discovery Modes. PR yields fewer typed connections than PO/CO but not always fewer constraints after reduction, as bounded-matching pruning removes constraints that often act as transitive bridges. On BPIC2014 (330 activities) CO reduces 12541 constraints by 84% to 1974, while PO/PR start from ∼6340 pruned constraints and reduce by only ∼60% to 2422/2573. Discovery took 40 ms to ∼20 s, with PR consistently the fastest (≤6 s).

6 Discussion & Conclusion

We presented a lossless transitive reduction and activity projection for OC-Declare existence models, with proven language equivalence. Averaged over the six datasets, the reduction yields >60% size improvement for complete discovery.

We recommend the lossless reduction as a default post-processing step after discovery: It is fast, language-preserving, and substantially compresses the model. Activity projection complements it when focusing on a subset of activities, and the bounded-matching filter [14] can add further (lossy) compaction.

The current scope covers OC-Declare existence constraints, the family supported by automatic discovery, and redundancy based on domination and transitivity. While we measure structural simplification with graph-based metrics adapted from DCR graphs [4], how this translates to cognitive complexity for OC-Declare remains an open question [11]. Extending the approach to negative and other constraint types [14] and integrating consistency-aware reduction in the object-centric setting [7] are further directions for future work.

Acknowledgments. The authors gratefully acknowledge the German Federal Ministry of Research, Technology and Space (BMFTR) and the state government of North Rhine-Westphalia for supporting this work as part of the NHR funding.

References

1. van der Aalst, W.M.P., Berti, A.: Discovering Object-centric Petri Nets. Fundam. Informaticae **175**(1-4) (2020)
2. van der Aalst, W.M.P., Li, G., Montali, M.: Object-Centric Behavioral Constraints. CoRR (2017)
3. Aho, A.V., Garey, M.R., Ullman, J.D.: The transitive reduction of a directed graph. SIAM J. Comput. **1**(2) (1972)
4. Andaloussi, A.A., Burattin, A., Slaats, T., Kindler, E., Weber, B.: Complexity in declarative process models: Metrics and multi-modal assessment of cognitive load. Expert Syst. Appl. **233** (2023)
5. Berti, A., Koren, I., Adams, J.N., Park, G., Knopp, B., Graves, N., Rafiei, M., Liß, L., Tacke genannt Unterberg, L., Zhang, Y., Schwanen, C.T., Pegoraro, M., van der Aalst, W.M.P.: OCEL (Object-Centric Event Log) 2.0 Specification. CoRR (2024)
6. Christfort, A.K.F., Rivkin, A., Fahland, D., Hildebrandt, T.T., Slaats, T.: Discovery of Object-Centric Declarative Models. In: ICPM. IEEE (2024)
7. Di Ciccio, C., Maggi, F.M., Montali, M., Mendling, J.: Resolving inconsistencies and redundancies in declarative process models. Inf. Syst. **64** (2017)
8. Di Ciccio, C., Mecella, M.: On the discovery of declarative control flows for artful processes. ACM Trans. Manag. Inf. Syst. **5**(4) (2015)
9. Fahland, D., van der Aalst, W.M.P.: Simplifying discovered process models in a controlled manner. Inf. Syst. **38**(4) (2013)
10. Gianola, A., Montali, M., Winkler, S.: Object-Centric Conformance Alignments with Synchronization. In: CAiSE. LNCS, vol. 14663. Springer (2024)
11. Haisjackl, C., Barba, I., Zugal, S., Soffer, P., Hadar, I., Reichert, M., Pinggera, J., Weber, B.: Understanding declare models: strategies, pitfalls, empirical results. Softw. Syst. Model. **15**(2) (2016)
12. Khayatbashi, S., Hartig, O., Jalali, A.: Transforming event knowledge graph to object-centric event logs: A comparative study for multi-dimensional process analysis. In: ER. LNCS, vol. 14320. Springer (2023)
13. Küsters, A., van der Aalst, W.M.P.: Rust4PM: A versatile process mining library for when performance matters. In: BPM (Demos). CEUR, vol. 3758 (2024)
14. Küsters, A., van der Aalst, W.M.P.: OC-DECLARE: discovering object-centric declarative patterns with synchronization. In: BPM. LNCS, vol. 16044. Springer (2025)
15. Maggi, F.M., Bose, R.P.J.C., van der Aalst, W.M.P.: A knowledge-based integrated approach for discovering and repairing declare maps. In: CAiSE. LNCS, vol. 7908. Springer (2013)
16. Maggi, F.M., Montali, M., Di Ciccio, C., Mendling, J.: Semantical vacuity detection in declarative process mining. In: BPM. LNCS, vol. 9850. Springer (2016)
17. Park, G., Adams, J.N., van der Aalst, W.M.P.: Conformance checking and performance analysis using object-centric directly-follows graphs. In: BPM (Forum). LNBIP, vol. 526. Springer (2024)