Enabling Object-Centric Process Mining from Time-Series

Abd Alrhman Abu Sbeit[1], Franco Cavadini[2], and Dirk Fahland[1]

[1] Eindhoven University of Technology, Eindhoven, Netherlands,
{a.a.m.s.abu.sbeit, d.fahland}@tue.nl
[2] GR3N SA, Lugano, Switzerland,
franco.cavadini@gr3n-recycling.com

Abstract. Object-Centric Process Mining (OCPM) requires discrete events that explicitly reference activities and the objects they affect. In industrial processes where physical materials move as batches from one station to another, such as in the chemical industry, behavior is recorded solely by low-level sensors as time-series data. Without activities and object identifiers being recorded, OCPM is inapplicable. This paper proposes a novel methodology to transform raw time-series sensor data into semantically meaningful, discrete events, and to infer objects and relations that conform to OCPM requirements. We apply this methodology to real-world sensor data from a polyethylene terephthalate (PET) chemical recycling process. Our results show that this transformation enables object-centric analysis of industrial processes, validated through expert feedback and alignment with real material flows.

Keywords: Object-Centric Process Mining, Time-Series Discretization, Inference

1 Introduction

Industrial environments generate large volumes of time-series data from sensors monitoring equipment like valves, pumps, vessels, and pipelines. These high-frequency signals reflect the evolving state of discrete actuators and continuous variables (e.g., flow, level, temperature), but they do not directly represent process-level events or object behavior. Traditional activity-centric process mining techniques fall short in capturing concurrent object movements and interactions. Object-Centric Process Mining (OCPM) [1] addresses this by modeling multiple interacting entities (e.g., material batches), but it requires discrete event data with associated objects.

OCPM relies on object-centric event data (OCED) composed of (O1) discrete events with timestamps and types, (O2) uniquely identified objects, and (O3) explicit relations between them. However, in our setting: (I1) data originates from high-frequency, noisy time-series (boolean and continuous), (I2) multiple sensors monitor parts of the same activity but no single sensor indicates its full extent, and (I3) object existence, movement, or processing is not explicitly recorded. The challenge is to derive (O1–O3) from input data exhibiting (I1–I3).

We focus on material handling processes with the following characteristics: (C1) handled objects are unrecorded (semi-)batches of material, (C2) materials reside at stations or move between them via defined connections (e.g., pipes), (C3) each connection carries at most one batch at a time, and (C4) batches can merge or split at stations. (C5) Movements are driven by equipment (e.g., pumps) whose operation is tracked by low-level sensors, providing input with characteristics (I1–I3). The goal of this paper is to reconstruct batch presence and movement (C1, C2, C4) as OCED (O1–O3), by exploiting other characteristics (C3) to enable OCPM techniques to be applied to such processes, complementing traditional manufacturing process analysis methods.

This gives rise to the following sub-problems: (P1) discretize noisy low-level time-series into event records per sensor, (P2) aggregate low-level events from related sensors into process-level events, (P3) infer objects from process-level events, and (P4) link events to inferred objects. Hence, the overall task is to infer (O1–O3) from data with properties (I1–I3), where events, objects, and relations are not explicitly present. P1 has been explored in prior work [1], [2], [3]. P2 has been studied for sequential low-level compositions [4], whereas we consider parallel event composition. P3 and P4 have been addressed in settings where some object references exist [5]. The novelty of our work wrt. P2-P4 lies in addressing a setting where no case identifiers or object identifiers exist at all, and in addressing P1-P4 in complete method.

Our methodology (Figure 1) begins by discretizing raw sensor signals into low-level operational events (e.g., transfer active/inactive) using mappings and debouncing techniques to reduce noise (Section 3.1). These low-level events are then integrated across sensors monitoring the same connection to identify consensus-based movement periods (Section 3.2), from which lifecycle events are inferred. This enables grouping low-level events into process-level transfer events between stations. Using event knowledge graphs (EKGs), we reconstruct object behaviors such as transfers, merges, and splits (Section 4), resulting in OCED suitable for OCPM tasks like discovery and conformance checking.

We validate our framework using real-world data from Gr3n, a chemical recycling company processing heterogeneous material batches across sensor-monitored stations. The resulting events accurately capture object flow and are confirmed by the use case representative (Section 5), forming the foundation of OCPM over their recorded time-series data.

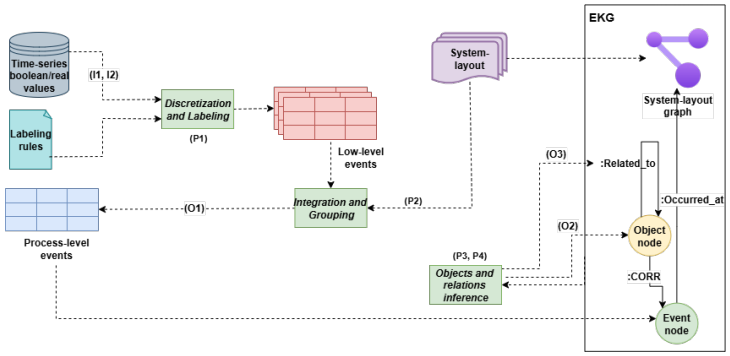


Fig. 1: Approach illustration.

2 Related Work

Transforming raw sensor data into discrete events is essential for enabling process mining, especially in object-centric settings. Traditional methods rely on a single-case perspective and cannot capture interactions between multiple entities. OCPM overcomes this by linking events to multiple objects, enabling the analysis of concurrent behaviors. OCPM assumes input in the form of OCED [6], which structures events, objects, and their relations, and is currently being standardized [7]. Our context requires modeling behavioral dependencies between different batches of material, including merge and split relations, which can be represented using EKGs [8]. An EKG is a labeled property graph where nodes (e.g., `:EVENT`, `:ENTITY`) and relationships (e.g., `:CORR`, `:DF`) carry labels and properties. EKGs can be extended to model event context and relations between entities [5], [9].

Sensor data, typically noisy and high-volume, must be abstracted into discrete events for analysis. Brzychczy et al. [2] emphasize segmentation and thresholding, while Garcia et al. [3] show how categorical intervals facilitate event-based reasoning. The standard approach involves segmenting time-series based on dynamics and labeling intervals via supervised learning [10] or rule-based methods [11]. We adopt a similar principle. At a higher level, discrete low-level events must be related to process-level activities. Janssen et al. [12] apply clustering to identify activity patterns, and Bayomie et al. [13] show how discretized sensor signals support manufacturing process mining and digital twin construction. Mannhardt et al. [14] detect activities using predefined low-level event sequences, but such sequential patterns are unavailable in our case, as low-level events occur concurrently across sensors without a known order. The problem of identifying high-level events from concurrent low-level events has also been investigated in [4]. However, their focus differs: they analyze how subsequences of low-level events group into high-level events that occur in parallel with other low-level events in the presence of case identifiers. In contrast, our goal is to determine which parallel low-level events should be consolidated into high-level events without the availability of case identifiers.

Problems P3 and P4, inferring objects and their relations, relate to techniques that handle incomplete event logs where only some events reference cases or objects. These approaches use process models, rules, or constraints to infer missing identifiers [15], [16], [5], or surrogate identifiers [17]. However, our setting lacks any object references in the abstracted events, meaning object existence itself must be inferred. Additionally, objects in our setting are unidentified material batches subject to transformation, merging, and splitting, requiring the discovery of both objects and their interrelations. EKG-based approaches [5] have shown that missing event-object links can be inferred by modeling the environment and applying logical rules. Similarly, Toosinezhad et al. [18] infer high-level event relations based on temporal and spatial overlap. We built on these insights to address the unique challenge of inferring both objects and their relations from events that lack explicit object information.

The contribution of this paper aims to bridge the gap between raw sensor data and high-level process insights, thereby encouraging the application of OCPM in industrial settings.

3 Transformation of Time-Series Data into Discrete Events

We work with raw sensor data that is noisy, redundant, and lacks semantic context, particularly when multiple sensors capture overlapping aspects of the same activity. To resolve this, we apply domain-specific interpretation rules combined with temporal filtering to extract meaningful operational changes.

This section presents the methodology for converting heterogeneous, high-frequency boolean and real-valued sensor streams into structured discrete event intervals. Section 3.1 addresses sub-problem P1 (Figure 1) by transforming individual sensor signals into low-level events. Section 3.2 addresses P2 by aggregating low-level events into process-level events.

3.1 Discretization Methodology

The discretization of time-series data is done through (a) determining, via domain knowledge, how to map a signal in the time-series to a discrete status, (b) segmenting the time-series into intervals holding the same status, and (c) filtering and debouncing noisy signals.

To solve P1, we build on existing approaches [2], [19], [11], [20]: we first segment a time-series signal into intervals of "temporal stability" and then assign each interval a label indicating the type of activity observed by the sensor.

Status Interpretation We use a rule-based approach to map a time-series signal to a status [19]. For discrete-value time-series, we simply map from a time-series value to a status, e.g., boolean value *false* and *true* indicating *Transfer Inactive* and *Transfer Active* respectively. For real-value time-series, we map a *value trend* to a status, e.g. values in a specific range (in a motor speed sensor) map to *Transfer Active*.

Construction of Transfer Periods Within our context (C1-C5), the goal is to detect intervals that reflect states such as active or inactive transport, open or closed valves, or high or low material levels, which correspond to concrete physical operations in the system under observation.

With a known interpretation of a time-series entry to a status, we segment the entire time-series into maximal continuous periods where all time-series entries map to the same status [11]. Table 1 presents the outcome of the discretization process applied to the time-series in Figure 2. The first interval corresponds to the continuous sequence of 'false' values prior to 09:50:22, while the second interval captures the continuous 'true' values observed between 09:50:22 and 10:39:05, as depicted in the figure.

Noise Filtering and Debouncing The output of this transformation is a set of low-level event records, each representing a time segment during which a specific sensor

Table 1: Structured transfer events extracted from raw sensor data

InstanceId	StartTimestamp	EndTimestamp	Status
4373	2024-12-19 08:02:12	2024-12-19 09:50:22	Inactive
4373	2024-12-19 09:50:22	2024-12-19 10:39:05	Active
4373	2024-12-19 10:39:05	2024-12-19 18:13:17	Inactive
4373	2024-12-19 18:13:17	2024-12-19 19:55:00	Active
4373	2024-12-19 19:55:00	2024-12-20 11:45:57	Inactive

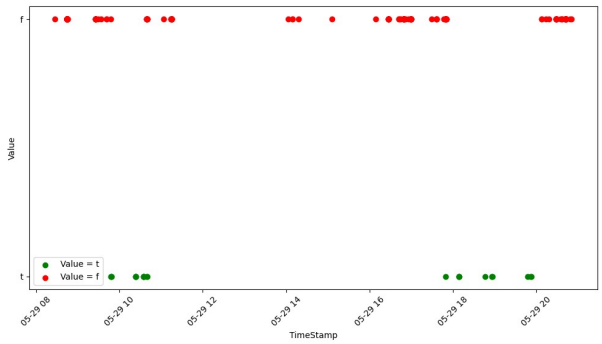


Fig. 2: Boolean time-series sensor recordings.

maintained a semantically interpretable state. Formally, each segment is represented as a tuple of the form:

$$\langle \texttt{sensorId}, T_{start}, T_{end}, \texttt{label} \rangle$$

where T_{start} and T_{end} define the temporal boundaries of an interval during which the sensor signal is interpreted as holding a stable semantic meaning (e.g., "Transfer Active" or "Transfer Inactive").

In cases where the status signal fluctuates rapidly within a short time interval, these variations are often attributed to electrical noise, sensor jitter, or transient physical behaviors that do not reflect meaningful process changes. Padala et al. [20] introduce a "refractory period" in event-based vision sensors, rejecting events that occur too closely in time to prior events at the same location—effectively debouncing high-frequency noise, we adopt this approach by merging two adjacent intervals with the same label (s, t_0, t_1, ℓ) (s, t_2, t_3, ℓ) into a larger interval (s, t_0, t_3, ℓ) whenever they are separated by a too short time-interval $t_2 - t_1 < k$ where parameter k is determined by analyzing historical sensor data and sensor signal behavior to identify the shortest meaningful signal changes, and it is validated by evaluating its impact on event detection accuracy in known scenarios.

3.2 Integrating Multi-Sensor Events for Inferring Object Movements

The generated events represent the perspective of individual sensors during material movement activities. Each such interval captures a specific signal (e.g., valve open, pump active) observed during transport, but in isolation, these sensor-level events provide only a partial view of the full material movement from one station to another.

We thus now address sub-problem P2 to reconstruct complete transport events—i.e., periods during which material reliably moves from station S1 to station S3—by integrating the partial perspectives from multiple sensors. These sensors may reflect various aspects of the same physical activity (e.g., valve opening, flow rate change, pressure drop) but are distributed and asynchronous.

Our approach proceeds in three steps:

Step 1: Integration of sensor data requires context information of the system layout. In our setting (C1-C5), we can group all sensors associated with the same physical

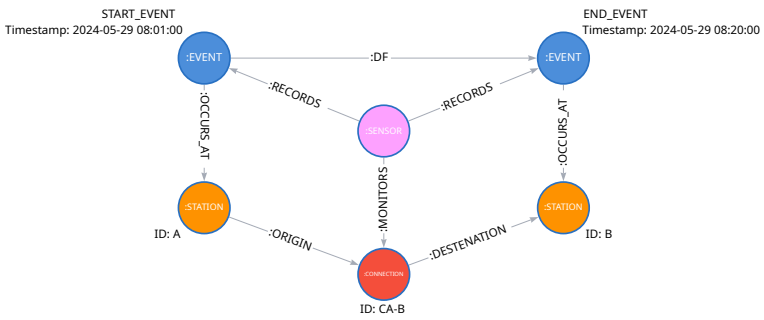


Fig. 3: EKG representation.

transport connection (e.g. CA-B from station A to B) using system topology that describes which sensor records activity for which connection between two stations.

Step 2: We model the known system topology by generating an EKG with `:STATION` nodes and `:SENSOR` nodes linked to the respective pipeline between stations. We model each pipeline from one station to another by a distinct `:Connection` node with `:ORIGIN` and `:DESTINATION` relations to the source and target station. This graph models the flow structure and sensor placements as shown in Figure 3.

Step 3: We now can consider all low-level event records identified in 3.1 of sensors related to the same connection C. We infer a process-level event at C as a material movement interval that is jointly explained by the sensors at C. Technically, we infer material movement intervals by identifying the longest time intervals during which a majority of the sensors assigned to the connection agree on observing activity. Specifically, for each connection, we query the event intervals of all associated sensors $S_1, ..., S_n$, and identify each maximal interval $[t_1, t_2]$ such that in every sub-interval $[t'_1, t'_2] \subseteq [t_1, t_2]$, at least n sensors report overlapping activity (i.e., have an interval $[t''_1, t''_2]$ overlapping $[t'_1, t'_2]$). The parameter n is set to be 65% of N, where N is the total number of sensors per connection, and adjusted dynamically in cases of sensor failure or missing data to reflect the number of available sensors. This approach provides robustness to sensor noise, jitter, and temporary outages. While a minority of sensor signals may be ignored in the presence of conflicting data, such trade-offs are acceptable in this context due to the physical constraints that reinforce correct material flow detection.

Finally, each inferred material movement interval $[t_1, t_2]$ at a `:CONNECTION` (red node) recorded via a `:SENSOR` (pink node) is transformed into two atomic `:EVENT` (blue nodes) with timestamps t_1 and t_2, representing the start and end of the movement, respectively. These nodes are connected by a `:DF` (directly follows) relation and linked to the relevant `:SENSOR` and the source and target `:STATION` (orange nodes), as illustrated in Figure 3. This abstraction enables reasoning about material flow as a coherent, process-level behavior rather than isolated sensor-level observations.

4 Objects and relations Inference

In Section 3, we generated a partial EKG that contains the inferred process `:EVENT` nodes linked to system topology. The system topology provides context information for the events that allows us to address P3 (inferring objects) and P4 (inferring relations)

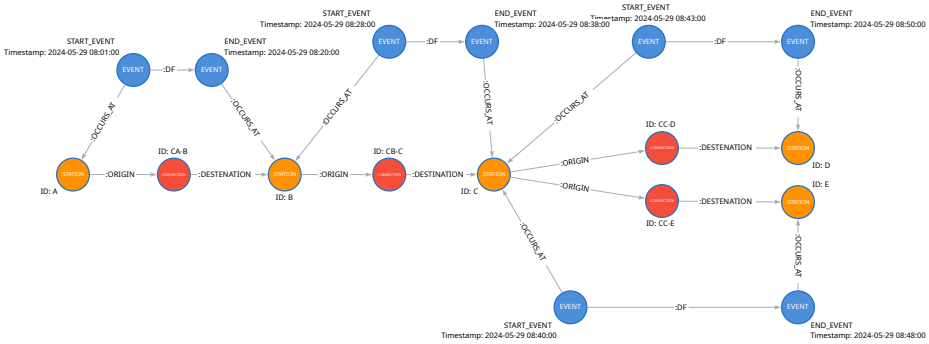


Fig. 4: EKG representing transfer events and stations based on segmented sensor data.

by exploiting the characteristics C1-C5 of ours setting, Figure 4 visualizes the situation serving as example for this section.

The problem is similar to the problem of inferring object identifiers in the context of physical process any location (i.e. in our setting any `:CONNECTION`) can hold at most one object at a given time. For this problem, Swevels et al. [5] identify three fundamental physical principles for inferring the presence and identity of physical entities within industrial systems. Building on these principles, we define a set of inference rules (IR1–IR5) to enrich the EKG with object nodes and their relations, based on the partial EKG. We first infer the existence of an object from a material transport event (IR1), to address P3. We then infer relations between transported materials as passing through a station (IR2), splitting materials at a station (IR3) and merging materials at a station (IR4). Finally, we infer directly-follows relations between events based on the material relations (IR5). We now explain each rule.

(IR1) From two `:EVENT` nodes e_1 and e_2 originating from the same `:SENSOR` group at a connection C from station A to station B we can infer that one batch of material has been moved from A to B via C. We can materialize this knowledge as a new `:BATCH` node (representing the material batch) correlated to e_1 and e_2.

Figure 5a illustrates the corresponding inference rule IR1 where the nodes and relations with solid edges indicate the left-hand side (LHS) of the rule (as a structure to be present in the EKG) and the nodes and relations with dashed edges indicate the right-hand side (RHS) of the rule (as the structure to be added to the EKG in relation to the LHS). Such rules can be directly expressed as graph queries over EKGs, e.g., in Cypher, see [5].

Applying the rule on the EKG detects all instances of the LHS, i.e., all material transport event pairs, and extends the EKG with the nodes of the RHS. However, each object (batch) node now is correlated only to two event nodes (start and end events), rather than tracking a single object across all the stations.

To model an end-to-end process flow showing transporting of material across stations, we have to infer relations between objects, for which we use directed `:RELATED_TO` relations for any two batches that appear time-wise sequentially at the same station, where the first batch finishes entering the station (labeled End), and the second batch starts leaving the station (labeled Start). Here, we have to distinguish three cases based on the system topology.

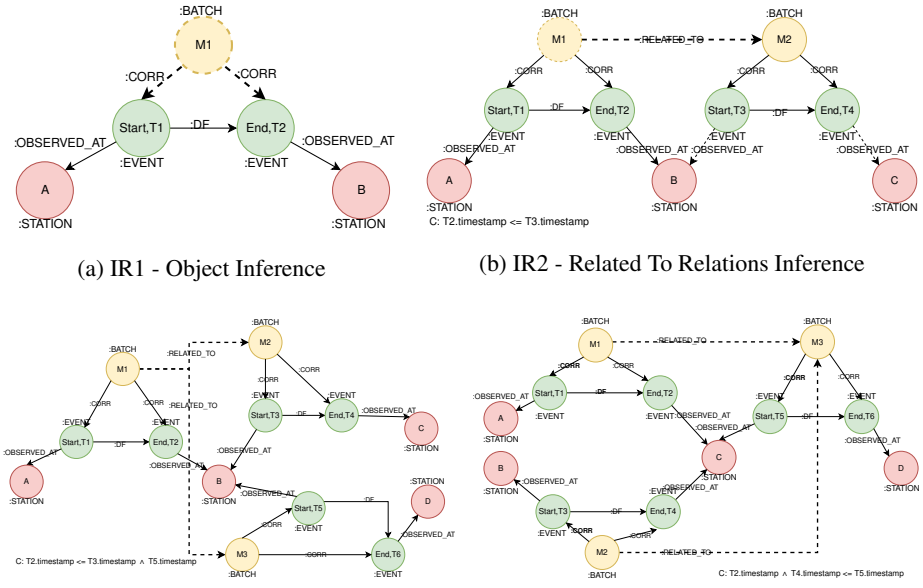
(IR2) In the most basic case, a batch M_2 is considered related to batch M_1 if both are correlated to events observed at the same station and the start time of M_2 is greater than or equal to the end time of M_1. This captures scenarios where materials are processed in sequence, such as continuous feeding, waiting, or processing of subsequent batches within the same unit. Figure 5b visualizes the rule IR2 which, next to the LHS and RHS has an additional condition that $T2.timestamp \leq T3.timestamp$.

(IR3) A batch M_2 is inferred to originate from a batch M_1, but does not comprise all material of M_1, if M_1 ends at a station from which multiple downstream batches M_2, M_3 emerge shortly thereafter, each directed toward different destination stations. Applying the rule for all batches leaving a station models the splitting of material into distinct sub-batches based on type or quantity. The rule compares timestamps to establish temporal succession, ensuring that the start time of each outgoing batch follows the end time of the shared incoming batch. Each resulting batch is then linked to its source via a `:RELATED_TO` relation in the EKG, enabling the reconstruction of diverging flow paths. Figure 5c illustrates this inference rule, next to the LHS and RHS has an additional condition that $T2.timestamp \leq T3.timestamp \wedge T5.timestamp$.

(IR4) A batch M_3 is inferred to result from the merging of multiple batches M_1, M_2, ..., M_i if these originate from different upstream stations and arrive at a common destination before the start of M_3. This rule captures the joining of material streams, such as in mixing or consolidation steps. The inference ensures that the start time of the resulting batch follows the end times of all contributing batches. Upon validation, each source batch is connected to the merged batch via a `:RELATED_TO` relation in the EKG, enabling backward traceability of material lineage. Figure 5d illustrates this inference rule, next to the LHS and RHS has an additional condition that $T2.timestamp \wedge T4.timestamp \leq T5.timestamp$.

(IR5) A `:DIRECTLY_FOLLOWS` relation is inferred between two transport events when the corresponding material batches are connected through any of the previously defined object-level relations (IR2–IR4). This rule captures the behavioral progression of the process, reflecting how material flows sequentially through the system. The resulting relation links consecutive object movements, enabling the reconstruction of end-to-end flow paths across the process network.

Figure 6 illustrates the EKG in Figure 4 after applying inference rules IR1–IR3 and IR5. The figure highlights (IR1) the creation of batch nodes (green), and the material flow between stations (orange). It also shows an example of (IR3) material splitting between stations S4, S5, and S6. Additionally, (IR2) related batches are connected via blue arrows, while (IR5) directly follows (DF) relationships between corresponding events are indicated by red arrows. These rules collectively allow us to construct an object-centric view of the physical process over time, aligning with the requirements of OCED as described by Fahland et al. [7]. Our case however demonstrates the need to explicitly model DF relations between events of different (related) objects as otherwise the data cannot describe behavior across multiple stations.



(a) IR1 - Object Inference

(b) IR2 - Related To Relations Inference

(c) IR3 - Material Splitting

(d) IR4 - Material Grouping

Fig. 5: Conceptual diagrams illustrating the defined inference rules.

5 Implementation and Validation

We validated the approach described in Sections 3 and 4 using a case study on an industrial polyethylene terephthalate (PET) recycling process operated by Gr3n, a company developing microwave-assisted alkaline hydrolysis for depolymerizing PET into ethylene glycol (EG) and disodium terephthalate (Na_2TA). This process is implemented at MADE, Gr3n's demonstration plant, which features a modular batch architecture of reactors, buffer tanks, separation units, and utilities. A dense sensor network monitors temperature, pressure, flow rates, valve states, motor commands, and other control flags. While the process is highly automated, the data infrastructure lacks structured semantic annotations such as batch identifiers, object tags, or event logs, limiting its use for high-level process analysis or OCPM.

We obtained a raw dataset covering several months of sensor readings from 218 sensors, calibrated to overcome clock drift, including both binary (e.g., valve open/closed, motor on/off) and real-valued (e.g., temperature, pressure, material level) signals. The dataset contains over three million timestamped records. Due to the proprietary nature of the project, neither the dataset nor the implementation code can be publicly available. The goal was to apply our methodology to extract discrete events from these raw time-series, infer object lifecycles, and construct an EKG enriched with object identifiers and relations between batches.

We implemented the time-series-to-event transformation using Python scripts based on labeling rules of Gr3n recordings. The EKG construction and object/relation inference were implemented using the open-source PromG library [21], available at `https://github.com/promg-dev`, specifying a semantic header to integrate system lay-

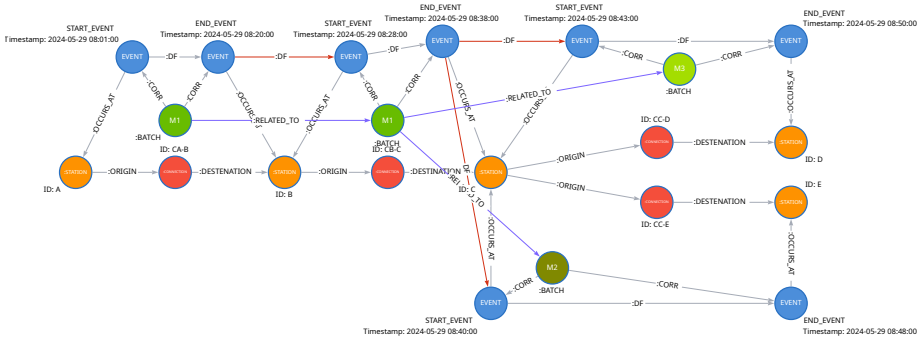


Fig. 6: EKG representation with after applying IRs.

out and events. Inference rules IR1–IR5 were encoded using PromG's inference engine. This process resulted in 4,146 low-level transportation events, grouped into 1,566 process-level events representing material transfers between stations. The EKG construction produced 1,566 material batches and 3,001 relations between them.

We validated the generated events and EKG with Gr3n's process owner, confirming that the extracted event sequences and inferred material flows align with known station connections, timing, and operational behavior. The object inference logic was validated as consistent with real-world batch movement scenarios, including splitting and merging behavior. The object-centric perspective accurately reflected lifecycle transitions, concurrency, and interactions across units.

Figure 7 shows a section of the inferred knowledge graph, with red and off-white nodes representing stations and connections, green nodes as events, and other colored nodes denoting different material batches and their relations. Merging of materials is visualized, such as M1 and M2 forming M3. Figure 8 further illustrates an end-to-end process execution as a partial order of batch movements, capturing lifecycle transitions across the system.

Runtime performance is practical, event discretization took 622.91 seconds due to the high-frequency input, while loading events into the EKG completed in 16.09 seconds, and object inference required 31.17 seconds. These results demonstrate the scalability and feasibility of the approach for large-scale industrial data.

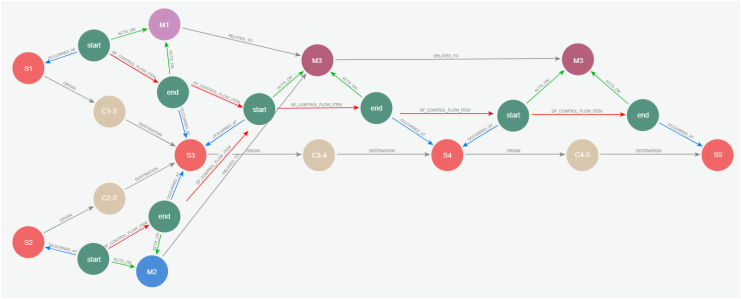


Fig. 7: Object-centric material flow graph inferred from sensor-based events.

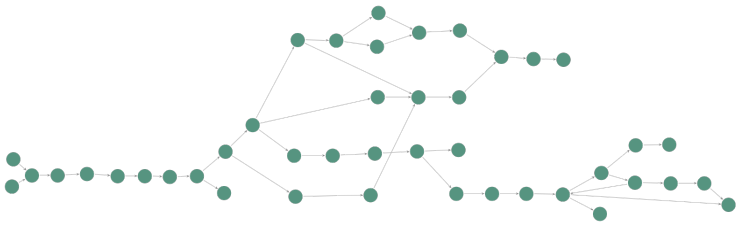


Fig. 8: End-to-end execution in Gr3n's EKG as a partial order of material batches.

6 Conclusion

This paper presented a novel methodology for transforming heterogeneous time-series sensor data lacking explicit events, objects, or relation, into OCED, which services as the foundation for applying OCPM.

The proposed methodology transforms heterogeneous time-series sensor data into OCED by integrating system topology at the sensor-level recordings and inferring unrecorded objects and relations. This enables OCPM in industrial contexts, such as chemical production, where object identities are not explicitly tracked. Validation in an industrial PET recycling case confirms feasibility. The approach currently applies to processes satisfying C1–C5. While this may sound limited, any manufacturing process handling physical material in distinct batches or heaps and where sensor records can be related to the movement of a distinct (unidentified) batch or heap.

Our focus on inferring objects and relations out of material transportation activities is driven by the scope of the available sensor data. While this data was sufficient for successfully inferring objects and relations, further process analysis may require additional data detecting other activities and processing events. Incorporating these into the EKG would follow similar steps but may require refined event detection techniques and adjustments to EKG construction and inference logic.

Lastly, while the use case representative feedback has been positive and indicates a promising direction for further analysis, the work lacks defined evaluation criteria that can be generalized to other use cases, due to the novelty of transforming low level sensor data into OCED suitable for OCPM.

Acknowledgments. The research underlying this paper was supported by AutoTwin EU GA n. 101092021.

References

1. W. M. P. van der Aalst, "Object-centric process mining: Dealing with divergence and convergence in event data," in *SEFM 2019*, ser. LNCS, vol. 11724, 2019, pp. 3–25.
2. E. Brzychczy, M. Aleknonytė-Resch, D. Janssen, and A. Koschmider, "Process mining on sensor data: A review of related works," *Knowledge and Information Systems*, 2025.
3. S. Garcia, J. Luengo, J. A. Saez, V. Lopez, and F. Herrera, "A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 734–750, 2013.

4. O. Dogan and M. de Leoni, "Parallelism-based session creation to identify high-level activities in event log abstraction," in *Process Mining Workshops*, J. De Smedt and P. Soffer, Eds. Cham: Springer Nature Switzerland, 2024, pp. 58–69.
5. A. Swevels, R. Dijkman, and D. Fahland, "Inferring missing entity identifiers from context using event knowledge graphs," in *Business Process Management*, 2023, pp. 180–197.
6. W. M. P. van der Aalst, "Object-centric process mining: Unraveling the fabric of real processes," *Mathematics*, vol. 11, no. 12, p. 2691, 2023.
7. D. Fahland, M. Montali, J. Lebherz, W. M. P. van der Aalst, M. van Asseldonk, P. Blank, L. Bosmans, M. Brenscheidt, C. di Ciccio, A. Delgado, D. Calegari, J. Peeperkorn, E. Verbeek, L. Vugs, and M. T. Wynn, "Towards a simple and extensible standard for object-centric event data (oced) – core model, design space, and lessons learned," 2024.
8. D. Fahland, *Process Mining over Multiple Behavioral Dimensions with Event Knowledge Graphs*. Cham: Springer International Publishing, 2022, pp. 274–319.
9. A. Swevels, D. Fahland, and M. Montali, "Implementing object-centric event data models in event knowledge graphs," in *ICPM 2023*, ser. LNBIP, vol. 503, 2023, pp. 431–443.
10. N. Tax, N. Sidorova, R. Haakma, and W. M. P. van der Aalst, "Event abstraction for process mining using supervised learning techniques," in *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016*, 2018, pp. 251–269.
11. M. L. van Eck, N. Sidorova, and W. M. P. van der Aalst, "Enabling process mining on sensor data from smart products," in *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, 2016, pp. 1–12.
12. D. Janssen, F. Mannhardt, A. Koschmider, and S. J. van Zelst, "Process discovery from sensor event data," in *ICPM 2020*, 2020.
13. D. Bayomie, K. Revoredo, S. Bachhofner, K. Kurniawan, E. Kiesling, and J. Mendling, "Analyzing manufacturing process by enabling process mining on sensor data," in *Proceedings of the ICPM Workshop on Digital Twin Engineering*, ser. CEUR Workshop Proceedings, vol. 3298, 2022.
14. F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. P. van der Aalst, and P. J. Toussaint, "From low-level events to activities - A pattern-based approach," in *BPM 2016*, ser. LNCS, vol. 9850, 2016, pp. 125–141.
15. D. Bayomie, A. Awad, and E. Ezat, "Correlating unlabeled events from cyclic business processes execution," in *Advanced Information Systems Engineering, CAISE 2016*, 2016, pp. 274–289.
16. D. Bayomie, C. D. Ciccio, and J. Mendling, "Event-case correlation for process mining using probabilistic optimization," 2022.
17. M. Pegoraro, M. S. Uysal, T.-H. Hülsmann, and W. M. P. van der Aalst, "Resolving uncertain case identifiers in interaction logs: A user study," 2022.
18. Z. Toosinezhad, D. Fahland, Ö. Köroglu, and W. M. P. van der Aalst, "Detecting system-level behavior leading to dynamic bottlenecks," in *ICPM 2020*, 2020, pp. 17–24.
19. A. Ramírez, N. Moreno, and A. Vallecillo, "Rule-based preprocessing for data stream mining using complex event processing," *Expert Systems*, vol. 38, no. 8, p. e12762, 2021.
20. K. Padala, S.-H. Ieng, C. Posch, and R. Benosman, "A noise filtering algorithm for event-based sensors using a refractory period," *Frontiers in Neuroscience*, vol. 12, p. 306, 2018.
21. A. Swevels, E. L. Klijn, and D. Fahland, "Object-centric process mining (and more) using a graph-based approach with promg," in *ICPM 2023*, ser. CEUR Workshop Proceedings, vol. 3648, 2023.