Learning Colored Petri Nets Using Object-Centric Event Data (OCED2CPN)

Wil M.P. van der Aalst
Process and Data Science (PADS), RWTH Aachen University
Aachen, Germany
wvdaalst@pads.rwth-aachen.de

***Abstract*—Traditional process mining assumes that each event is related to precisely one case. This can be compared to mainstream modeling notations using Petri nets (in particular WF-nets), BPMN diagrams, directly-follows graphs, flowcharts, or UML activity diagrams. These notations describe the life-cycle of a process instance (often called case in process mining). Currently, we see an uptake of Object-Centric Process Mining (OCPM), where events can refer to any number of objects and where objects can be related. OCPM includes process discovery starting from Object-Centric Event Data (OCED) to produce process models that describe different object types in a single diagram. Since Colored Petri Nets (CPNs) have been around for decades and have been widely used for modeling, verification, and simulation, CPNs are an obvious target model for OCPM. The transition from traditional process mining to OCPM can be compared with the transition from classical Petri nets to CPNs. However, it turns out to be very difficult to discover arbitrary CPNs. This keynote paper summarizes what has been done before and why it is challenging to discover CPNs. However, starting from OCED, subclasses of CPNs can be discovered (which we refer to as OCED2CPN). These insights are also relevant for conformance checking and forward-looking forms of process mining starting from OCED.**

***Index Terms*—Colored Petri Nets, Object-Centric Process Mining, Object-Centric Event Data, Process Mining**

I. Introduction

Process-mining software provides unique opportunities to improve processes in a data-driven manner. Process mining starts from event data that need to be extracted from information systems (e.g., SAP, Oracle, and Salesforce). These data are used to discover the actual processes, check compliance, diagnose problems, predict performance, and automatically take action [1], [2]. There are over 40 commercial offerings of process mining software and analyst firms like Garner now consider this to be a new and substantial category of software (see, e.g., the Gartner Magic Quadrant [3]). Starting point are *event data* where each *event* should have a *timestamp* and an *activity* label. In traditional process mining, it is also assumed that each event refers to *precisely one case*. This simplifying assumption helps to get started quickly. Also when people model processes by hand, they typically describe the lifecycle of a single case, i.e., most process models describe the ordering of of activities seen from the viewpoint of a single case (e.g., an order, an application, a patient, or a loan). Although the simplifying assumption has many advantages, it also creates some problems. In reality, events may refer to many objects. For example, a "place order" event may refer to one order, five items, and one customer, and a "deliver package" event may refer to one package, three items, and one customer. The package may contain items of different orders and items of one order may end up in multiple packages. It is impossible to pick a case notion that captures reality well. Picking orders, items, or packages as case identifiers lead to distorted views on a more complex reality [4], [5].

Supported by the Alexander von Humboldt (AvH) Stiftung and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC Internet of Production, 390621612.

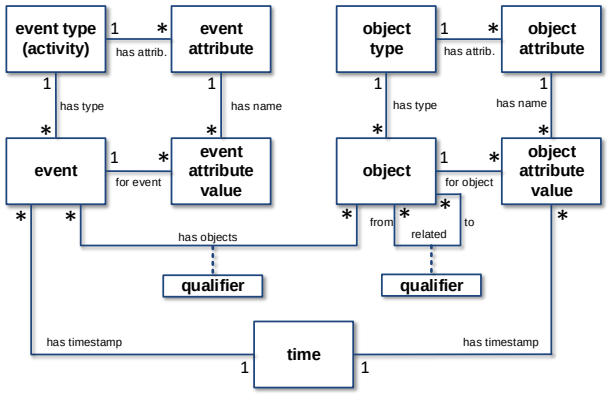

Fig. 1. Object-Centric Event Data (OCED) are composed of typed events and typed objects in a many-to-many relationship.

Object-Centric Process Mining (OCPM) takes a more holistic and more comprehensive approach to process analysis and improvement by considering multiple object types and events that involve any number of objects. For OCPM, we extract *Object-Centric Event Data* (OCED) described by the meta model in Figure 1 [5]. An event may be related to any number of objects and an object may be related to any number of events. Cases in traditional process mining can be viewed as objects which the additional requirements that (1) there is just one type (i.e., the selected case notion) and (2) events need to refer to precisely one object (i.e., the case). Both events and objects are *typed*. We often use the term *activity* to refer to an *event type*. There may be many events having the same type. Events also have a timestamp and are considered atomic. To model activity instances with an explicit duration, one needs to record a start and complete event. Examples of object types are orders, items, customers, suppliers, invoices,

resources, locations, patients, students, loans, etc. Events and objects can have any number of additional attributes. Event attribute values do not have an explicit timestamp, because the corresponding event has a timestamp. Object attribute values have a timestamp because for the same object the value of an attribute may change over time (e.g., price or temperature). Unlike traditional event data, OCED also allows for *Object-to-Object* (O2O) relationships next to the *Event-to-Object* (E2O) relationships. As Figure 1 shows, both O2O and E2O relations can be qualified (e.g. for a meeting event we can differentiate between objects of type person, e.g., participants, presenters, and the chair). Although the meta model in Figure 1 is rather simple, it is very powerful and generic. All objects and events of an organization can be stored systematically using this meta model.

Using OCPM, it is possible to view all operational activities from any perspective while establishing a single source of truth. It allows organizations should steer away from system-specific event logs (OCED are intended to be system agnostic). It is possible to "flatten" OCED by promoting one object type to become the selected case notion. This way it is possible to use all existing process mining tools and algorithms. However, when doing this, one should be aware of the well-known convergence and divergence problems [4], [5]. Events referring to multiple objects of the type selected as the case notion need to be replicated, possibly leading to unintentional duplication (called *convergence*). The replication of events can thus lead to misleading diagnostics. After flattening, there may be multiple events that refer to the same case and activity, but that differ with respect to one of the non-selected object types (called *divergence*). As a result, events referring to different objects of a type not selected as the case notion become indistinguishable. These are the usual problems one is confronted with when extracting traditional event data directly from the source systems. However, then it is hidden and also more time-consuming because one needs to redo the extraction when new questions emerge and the view changes. This is inflexible and prevents reuse. Using OCED, there is no need to extract the data when changing the viewpoint. This allows for flexibility using "on demand" process-mining views. OCPM also provides novel and valuable improvement opportunities for problems that live at the intersection points of processes and organizational units [5]. However, existing process mining techniques need to be extended to relax the single-case assumption. To this end, we developed techniques to discover Object-Centric Petri Nets (OCPNs) [6], [7]. In OCPNs, each place refers to an object type. This leads to the main question addressed in this keynote paper: *How does OCPM relate to Colored Petri Nets?*

Colored Petri Nets (CPNs) are an established formalism based on classical Petri nets to model and analyze concurrent processes involving different types of objects [8]–[10]. In classical Petri nets, objects are represented by tokens, but these are indistinguishable. Therefore, CPNs associate data values to tokens such that they become "colored" and distinguishable. Places are typed using so-called color sets. These color sets can be compared to object types in OCED. Given the similarities between events in OCED and transition occurrences in a CPN and the similarities between typed objects in OCED and colored tokens in a CPN, we elaborate on the relation between CPNs and OCPM. We will show that it is possible to discover subclasses of CPNs starting from OCED. We refer to such techniques as *OCED2CPN*. However, it is impossible to discover arbitrary CPNs. By discussing the relationship between CPNs and OCPM, we provide interesting insights guiding the further development of the process-mining discipline.

The remainder is organized as follows. Section II briefly introduces Colored Petri Nets (CPNs) using a few examples. Section III summarizes existing discovery approaches to learn CPNs using traditional event logs focusing on a single case notion. Section IV shows the challenges when trying to learn arbitrary CPNs. Section V sketches how to discover a restricted class of CPNs based on OCED. Section VI concludes the paper.

II. Colored Petri Nets

Petri nets are the oldest and still best-known formalism to model and analyze concurrent processes [11], [12]. A classical Petri net is a bipartite directed graph composed of *transitions* (represented by squares) and *places* (represented by circles). A place may contain *tokens* (shown as black dots). A *marking* is a distribution of tokens over places and represents the state of the process. A transition is enabled when each of the input places contains a token. An enabled transition may occur (called firing), removing a token from each input place and producing a token for each output place.

In a classical Petri net, tokens are indistinguishable (so-called "black tokens"). For many applications, it is meaningful to attach a value to a token. Such tokens are called *colored* tokens because now they can be distinguished based on their values. This leads to so-called *Colored Petri Nets* (CPNs) [8], [9] supported by tools such as *CPN Tools* (cf. cpntools.org). We use two small examples to explain some of the basics of CPNs.

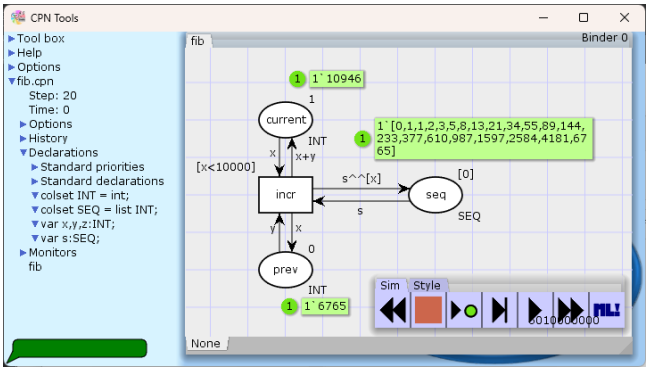

Fig. 2. CPN model to generate all Fibonacci numbers below 10000.

Figure 2 shows a CPN in *CPN Tools* generating a prefix of the Fibonacci sequence which is a sequence in which each

number is the sum of the two preceding ones. The model generates all Fibonacci numbers below 10000, i.e., 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, and 6765. For example, 6765 is the sum of the two previous numbers: 2584 and 4181. The places *prev* and *current* represent the two last numbers. Initially, these contain the numbers 0 and 1 respectively. Place *seq* contains a token that corresponds to a sequence of numbers. Initially, the value of the token in place *prev* is the sequence containing the first number in the sequence (i.e., 0). Transition *incr* consumes a token with value y from place *prev*, a token with value x from place *current*, and a token with value s (which is a sequence) from place *seq*. The transition produces a token with value x for place *prev*, a token with value $x + y$ for place *current*, and a token for place *seq* (its value is the sequence composed of the old sequence extended with x). Transition *incr* has a guard stating that x needs to be smaller than 10000. Figure 2 shows in green the marking after 20 transition occurrences. Place *seq* indeed contains a token with the Fibonacci sequence until 6765. Although the CPN in Figure 2 has little to do with operational processes, it reveals some of the basic concepts. Note that each place is typed by a color set. The places *prev* and *current* are of type *INT* (see the color set declaration on the left-hand side). Place *seq* is of type *SEQ* which is a list of integers. The arc inscriptions (e.g., $x+y$ on the arc connecting transition *incr* to place *current*) specify how consumed tokens are related to produced tokens.

Figure 3 shows another *CPN Tools* screenshot. There are three color sets: *Order* (to model orders represented by an integer), *Resource* (to model resources represented by a string), and *OrdRes* (the combination of an order and resource). The left part models the creation of orders. Transition *create order* consumes a token from place *counter* with value *ord* and returns a token with value *ord*$+1$ to the same place with a delay sampled from a negative exponential distribution having an expected value of 5 time units. All places are now timed. Transition *create order* also produces a token for place *o1* representing a new order and has a guard ensuring that precisely 100 orders are created. For each order, two production steps need to be performed. Transition *start assembly* models the start of the assembly process which requires a resource from place *free1*. Transition *complete assembly* models the end of the assembly process and returns a resource to place *free1*. Note that a token in place *or1* models the combination of an order and a resource. Transition *start painting* models the start of the painting process and transition *complete painting* models the end. Note that there are three resources for assembly (Pete, Mary, and John) and two resources for painting (Kate and Sue). Simulating the process to the end results in a marking with 100 tokens in place *o5* modeling the completed orders.

The last example CPN is shown in Figure 4. This example is relevant for object-centric process mining where a variable number of objects may be involved in an event. The CPNs shown in Figures 2 and 3 consume a fixed number of tokens. This is not the case for transitions *unpack* and *pack* in Figure 4. There are two color sets: *Item* and *Items* (to present a collection of items). Place *init* contains three example item sets. Therefore, transition *unpack* will occur three times. Transition *unpack* produces one token for place *inprod* and a variable number of tokens for place *i1*. Note that the arc inscription *is* on the arc from *unpack* to *i1* is of type *Items*, but place *i1* is of type *Item*. Hence, if *is* represents a collection of four items, then one token is produced for *inprod* and four tokens are produced for *i1*. Transitions *check1* and *check2* both consume and produce one token of type *Item*. Transition *pack* consumes one token from place *inprod* and a variable number of tokens from place *i3*. Actually, the items consumed from *i3* need to match the value of the token consumed from *inprod*. If the token is consumed from *inprod* represents a collection of four items, then precisely these four tokens need to be consumed from *i3*.

Figures 2, 3, and 4 illustrate that CPNs can be used to describe any discrete dynamic system. However, we would like to relate this to process mining: *Is it possible to discover CPNs from the event data in information systems?* To answer this question, we first consider traditional event data using a single case notion.

III. Learning Colored Workflow Nets

Figure 1 describes Object-Centric Event Data (OCED). However, most process mining techniques do not allow for multiple object types. In the classical setting, each event relates to precisely one case and a process model describes the lifecycle of individual cases in isolation. Each case is described by a sequence of events. In terms of Petri nets this relates to the class of *WorkFlow nets* (WF-nets) [13]. WF-nets are related to notations used in industry such as the BPMN (Business Process Model and Notation) standard [14]. Most process mining techniques focus on process models comparable WF-nets (e.g., BPMN, DFGs, process trees, and activity diagrams). After introducing CPNs, we can explain the limitations of WF-nets using the example in Figure 5.

The WF-net in Figure 5 describes the handling of request for compensation within an airline (the example is taken from [1]). Each case corresponds to a request. The process starts with activity *register request* and ends with activity *pay compensation* or activity *reject request*. There are two types of examination. One of these is done concurrent to checking the ticket. Then a decision is made. It is also possible to reinitiate the request. All the places in Figure 5 are of type *Case*, i.e., a token always refers to a case. Note that activity *register request* produces two tokens with the same identifier. Activity *decide* consumes two tokens that need to refer to the same case. WF-nets have a clear starting point and a clear end (here the source place *start* and the sink place *end* respectively). In a WF-net, cases flow from start to end which corresponds to the fact that in the event data each case is described by a sequence of events. Note that in Figure 5 we could drop the arc inscriptions and color sets of places because they are fixed anyway (i.e., *c* and *Case* respectively).

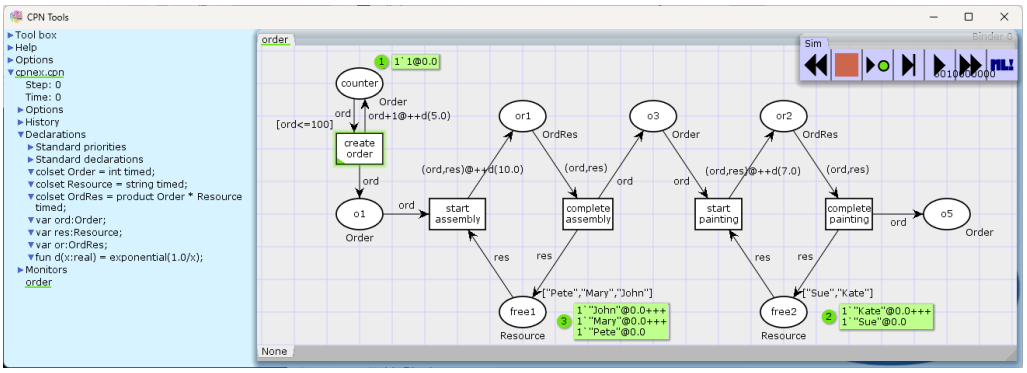

Fig. 3. CPN model modeling a production process with two steps: (1) assembly and (2) painting.

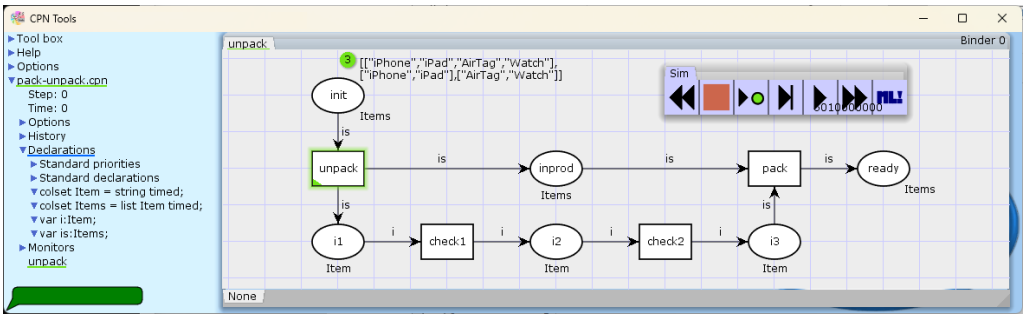

Fig. 4. CPN model showing that a multiset of tokens can be consumed or produced.

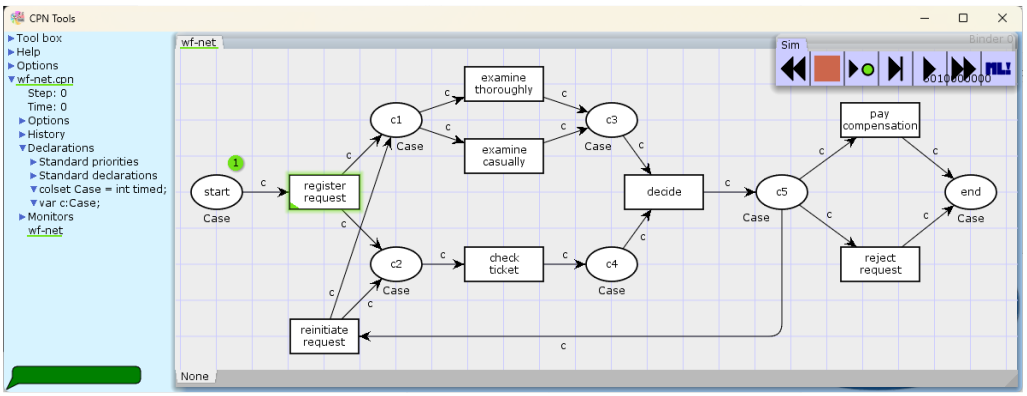

Fig. 5. A WF-net represented as a CPN with one color set *Case*. Note that all places have the same type and all arc inscriptions refer to case *c*.

A range of process discovery approaches using the "single-case assumption" have been proposed [2]. These can be grouped in "bottom-up" approaches like the Alpha algorithm and region-based techniques [1], [15]–[19] and 'top-down" approaches like the inductive mining techniques [20], [21]. See [22] for a recent survey of process discovery techniques.

Note that we focus here on control-flow only. However, we can add the other perspectives easily. Each event has a timestamp and the differences between two subsequent events for the same case can be used to learn the time distribution. Consider for example place *c1* in Figure 5. We can measure the time difference between *register request* or *reinitiate request* putting a token in *c1* and *examine thoroughly* and *examine causally* removing a token from *c1*. Using this information, we can add time distributions to the CPN model. It is also possible to add additional attributes to events and cases. In [23], it is shown how the different perspectives can be combined. The implementation described in [23] shows how *ProM* can be used to discover the control-flow and then add decision rules, resources, probabilities, and time. The result is a CPN model that can be simulated using *CPN Tools*.

This earlier work shows that it is possible to discover CPNs from event data under the assumption of a specific case notion. However, just like in Figure 5, tokens need to refer to cases rather than arbitrary objects. There are no relations between different objects (i.e., cases) and each transition occurrence

refers to one case instead of a collection of objects.

IV. Learning Arbitrary CPNs

As [23] convincingly shows, it is possible to discover CPNs from traditional event logs covering control-flow, data-flow, resources, probabilities, and time. However, it is impossible to discover arbitrary CPNs. To explain this, let us try to understand what it means to be able to discover the WF-net shown in Figure 5. For simplicity, we limit ourselves to control-flow. Think now of the following *rediscoverability* experiment. Simulate the process model and create an event log that has some level of completeness. Since the model has a loop, we cannot expect to see all possible traces. However, we can assume that we see all directly-follows relations (i.e., if activity a_1 can be followed by activity a_2, we should see it at least once). The event log can be seen as a multiset of traces where each trace is a sequence of activities. Based on such information, we would like to rediscover the WF-net shown in Figure 5 or at least a WF-net that has the same behavior (e.g., modulo renaming places). The place names are not relevant and the arc inscriptions and color set are fixed. All the classical discovery algorithms mentioned before [15]–[21] will be able to rediscover the WF-net. However, Figure 5 is not an arbitrary CPN and we carefully represented the event log as a multiset of traces.

Now consider Figures 2, 3, and 4. What would be suitable event logs for these three models? If transition occurrences correspond to events, we could describe each event by the transition name, the time at which the transition fired, and the values of the variables used for the arc inscriptions. For example, if we abstract from time, the first transition occurrence of the CPN in Figure 2 could be described by $incr(x = 1, y = 0, s = [0])$. The next four transition occurrences are: $incr(x = 2, y = 1, s = [0, 1])$, $incr(x = 3, y = 2, s = [0, 1, 2])$, $incr(x = 5, y = 3, s = [0, 1, 2, 3])$, and $incr(x = 8, y = 5, s = [0, 1, 2, 3, 5])$. This triggers three challenges: (1) it is not reasonable to assume that this information is stored in existing information systems, (2) it is impossible to directly observe the markings (i.e., tokens in places), and (3) it is impossible to learn arbitrary functions relating the different variables (note that any function can be defined and used in the CPN). To address the second challenge, more information needs to be stored in the event log. This is unrealistic because this would imply that the event log is already showing parts of the model and the model is not really learned. When learning form data, one always needs to assume a target representation, e.g., a linear function in linear regression or a decision tree in decision tree learning. In a CPN, functions can be arbitrarily complex, e.g., the Ackermann function could have been applied to x and y in Figure 2.

To properly define a discovery task, we need to fix the input representation (e.g., a multiset of traces) and the class of possible target models (e.g., WF-nets). We cannot assume that the input representation reveals the underlying structure of the model (e.g., places or arc inscriptions). Also, we need to limit the class of models. Therefore, we picked OCED and Object-Centric Petri Nets (OCPNs) as a starting point [6], [7].

V. Learning Object-Centric Petri Nets

As discussed, it is impossible to discover arbitrary CPNs. Therefore, it makes sense to return to the OCED meta model in Figure 1. It is natural to think of events as transition occurrences. It is also natural to think of objects as tokens in places. We assume that each place has a type (like in CPNs) and corresponds to an object type. We also assume that transitions do not create, merge, or destroy objects. Objects are created by the environment. This limits the possibilities and results in so-called *Object-Centric Petri Nets* (OCPNs). In an OCPN each place corresponds to an object type. There are two types of arcs: *normal arcs* consuming and producing one token (i.e., object) and *variable arcs* consuming and producing a variable number of objects. For variable arcs, cardinalities can be specified (at least one, at most five, etc.)

Figure 6 shows an OCPN represented as a CPN. There are three object types (*Order*, *Item*, and *Package*) and six event types (*place order*, *pick item*, *send invoice*, *pack items*, *receive payment*, and *deliver package*). The arcs with the arc inscription *items* are variable. For example, an event of type *place order* may refer to multiple items. Events of type *pack items* and *deliver package* may also refer to multiple items. An OCPN is deliberately "underspecified" and only provides typing and cardinality constraints. Objects of different types are related (e.g. a order refers to items) but this is in the data and does not need to be specified. It is impossible to provide more details here. Therefore, we refer to [6] where the approach is described. This discovery approach was also implemented in our open-source tools, e.g., the "OCELStandard" package in *ProM* (promtools.org), the *OC-PM* tool (ocpm.info), and *Object-Centric Process Insights* (ocpi.ai). *Celonis Process Sphere* uses similar ideas and was the first commercial implementation of OCPM [24].

VI. Conclusion

In this keynote paper, we related Colored Petri Nets (CPNs) to Object-Centric Process Mining (OCPM). CPNs, and related models such as predicate/transition nets [25], have been around for over 40 years [8], [9]. Various tools and analysis techniques have been developed for CPNs. When combining Petri nets with different types of objects, one naturally ends up with CPNs. However, CPNs have been mostly considered for modeling, verification, and simulation. There are very few approaches to discover CPNs and [23] is a notable exception. However, also this paper does not consider multiple types of objects and assumes a single case notion. Given the uptake of OCPM it makes sense to revisit the relation. What is the class of CPNs we can discover using Object-Centric Event Data (OCED) as defined by the meta model in Figure 1?

Answering this question is relevant as OCPM provides important advantages. Using OCPM data extraction is configured only once and does not need to be adapted when changing viewpoints or answering new questions. Redundancy

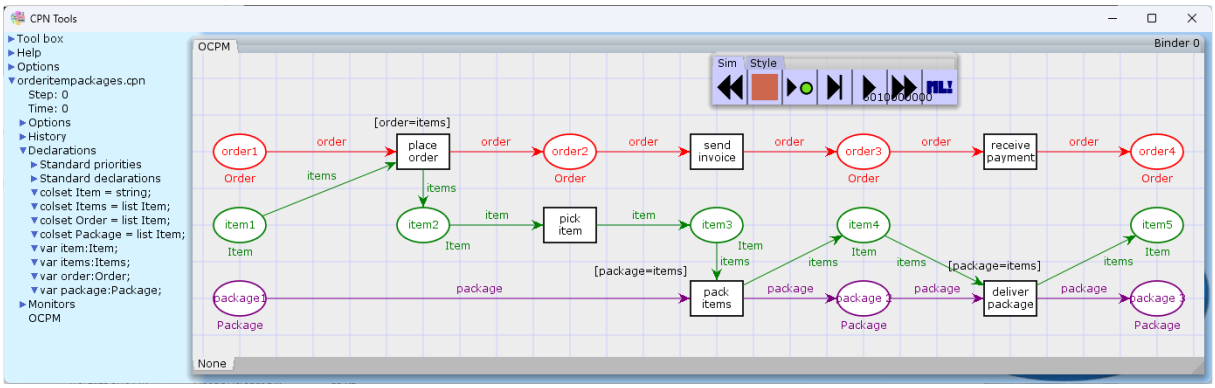

Fig. 6. An Object-Centric Petri Net (OCPN) represented as a CPN. The model has three object types: *Order*, *Item*, and *Package*.

is avoided because there is no need to create a data model per process. This provides more flexibility and makes data extraction less time-consuming. Using OCPM interactions between different types of objects can be analyzed. Many performance problems involve multiple interrelated objects. For example, in an assemble-to-order process, customer orders may be delayed due to unreliable suppliers for selected parts or the lack of production or transportation capacity. Often multiple processes are competing for resources. Therefore, the scope of analysis should not be limited to one type of objects. OCPM also helps to transition from "data push" to "data pull" using system-agnostic data models. Ideally, one would like to record events and objects at the business level and not at the database level. Distortions such as convergence and divergence are avoided by staying close to reality and it is possible to view events and objects from any angle without going back to the source systems.

References

[1] W. van der Aalst, *Process Mining: Data Science in Action*. Springer-Verlag, Berlin, 2016.

[2] W. van der Aalst and J. Carmona, Eds., *Process Mining Handbook*, ser. Lecture Notes in Business Information Processing. Springer-Verlag, Berlin, 2022, vol. 448.

[3] M. Kerremans, K. Iijima, A. Sachelarescu, N. Duffy, and D. Sugden, "Magic Quadrant for Process Mining Tools, Gartner Research Note GG00774746," 2013, www.gartner.com.

[4] W. van der Aalst, "Object-Centric Process Mining: Dealing With Divergence and Convergence in Event Data," in *Software Engineering and Formal Methods (SEFM 2019)*, ser. Lecture Notes in Computer Science, P. Ölveczky and G. Salaün, Eds., vol. 11724. Springer-Verlag, Berlin, 2019, pp. 3–25.

[5] ——, "Object-Centric Process Mining: Unraveling the Fabric of Real Processes," *Mathematics*, vol. 11, no. 12, p. 2691, 2023.

[6] W. van der Aalst and A. Berti, "Discovering Object-Centric Petri Nets," *Fundamenta Informaticae*, vol. 175, no. 1-4, pp. 1–40, 2020.

[7] A. Berti and W. van der Aalst, "OC-PM: Analyzing Object-Centric Event Logs and Process Models," *International Journal on Software Tools for Technology Transfer*, vol. 25, no. 1, pp. 1–17, 2023.

[8] K. Jensen, *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, ser. EATCS monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1992.

[9] K. Jensen and L. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Springer-Verlag, Berlin, 2009.

[10] W. van der Aalst and C. Stahl, *Modeling Business Processes: A Petri Net Oriented Approach*. MIT Press, Cambridge, MA, 2011.

[11] J. Desel and J. Esparza, *Free Choice Petri Nets*, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK, 1995, vol. 40.

[12] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, April 1989.

[13] W. van der Aalst, K. van Hee, A. Hofstede, N. Sidorova, H. Verbeek, M. Voorhoeve, and M. Wynn, "Soundness of Workflow Nets: Classification, Decidability, and Analysis," *Formal Aspects of Computing*, vol. 23, no. 3, pp. 333–363, 2011.

[14] OMG, "Business Process Model and Notation (BPMN), Version 2.0.2," Object Management Group, www.omg.org/spec/BPMN/, 2014.

[15] W. van der Aalst, A. Weijters, and L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004.

[16] A. Augusto, R. Conforti, M. Marlon, M. La Rosa, and A. Polyvyanyy, "Split Miner: Automated Discovery of Accurate and Simple Business Process Models from Event Logs," *Knowledge Information Systems*, vol. 59, no. 2, pp. 251–284, 2019.

[17] R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser, "Process Mining Based on Regions of Languages," in *International Conference on Business Process Management (BPM 2007)*, ser. Lecture Notes in Computer Science, G. Alonso, P. Dadam, and M. Rosemann, Eds., vol. 4714. Springer-Verlag, Berlin, 2007, pp. 375–383.

[18] J. Carmona, J. Cortadella, and M. Kishinevsky, "A Region-Based Algorithm for Discovering Petri Nets from Event Logs," in *Business Process Management (BPM 2008)*, 2008, pp. 358–373.

[19] M. Solé and J. Carmona, "Process Mining from a Basis of State Regions," in *Applications and Theory of Petri Nets 2010*, ser. Lecture Notes in Computer Science, J. Lilius and W. Penczek, Eds., vol. 6128. Springer-Verlag, Berlin, 2010, pp. 226–245.

[20] S. Leemans, D. Fahland, and W. van der Aalst, "Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour," in *Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2013)*, ser. Lecture Notes in Business Information Processing, N. Lohmann, M. Song, and P. Wohed, Eds., vol. 171. Springer-Verlag, Berlin, 2014, pp. 66–78.

[21] ——, "Scalable Process Discovery and Conformance Checking," *Software and Systems Modeling*, vol. 17, no. 2, pp. 599–631, 2018.

[22] A. Augusto, R. Conforti, M. Dumas, M. Rosa, F. Maggi, A. Marrella, M. Mecella, and A. Soo, "Automated Discovery of Process Models from Event Logs: Review and Benchmark," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 4, pp. 686–705, 2019.

[23] A. Rozinat, R. Mans, M. Song, and W. van der Aalst, "Discovering Simulation Models," *Information Systems*, vol. 34, no. 3, pp. 305–327, 2009.

[24] W. van der Aalst, "Object-Centric Process Mining: The Next Frontier in Business Performance," 2023, celon.is/OCPM-Whitepaper.

[25] H. Genrich and K. Lautenbach, "The Analysis of Distributed Systems by means of Predicate/Transition-Nets," in *Semantics of Concurrent Compilation*, ser. Lecture Notes in Computer Science, G. Kahn, Ed., vol. 70. Springer-Verlag, Berlin, 1979, pp. 123–146.