Multi-perspective path semantics in process mining based on causal process knowledge

Lukas Pfahlsberger[1,*], Christoffer Rubensson[1], Steven Knoblich[1], Maxim Vidgof[2] and Jan Mendling[1]

[1]Humboldt-Universität zu Berlin (HU Berlin), Rudower Chaussee 25, 12489 Berlin-Adlershof, Germany
[2]Wirtschaftsuniversität Wien (WU), Welthandelsplatz 1, 1020 Wien, Austria

Abstract

Process mining allows process analysts to investigate business processes with the help of algorithms and event log data. To better identify and understand inefficiencies in discovered process models, various visualization techniques have been proposed to enhance these models with further information, such as displaying the duration of execution time between activities using sequential color schemes or integrating statistical metrics into the model through textual annotations. However, it remains a challenge for analysts to identify interesting behavioral patterns in directly follows graphs. Consequently, this may lead process analysts to draw incorrect conclusions or be unable to identify the root causes for answering their analytical questions. This paper proposes a novel set of path semantics based on causal knowledge. We further examine how several combined path semantics, referred to as pattern types, may provide analysts with additional information on the underlying behavior. By examining an order-to-cash process in the real world, we demonstrate the usefulness and additional benefits of these path semantics for process analysts.

Keywords

Process mining, visual analytics, path semantics, causal process knowledge, directly follows

1. Introduction

Over the last two decades, a plethora of different techniques, methods, and approaches for visually representing business processes discovered from event-log data has been developed to support analysts in making better and faster decisions [1]. However, an essential part of these proposed process representations hardly differs in the semantic meaning of the visual components. In particular, the semantics of the paths are almost without exception limited to a single meaning, namely, a directly follows relationship.

Companion Proceedings of the 16th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling and the 13th Enterprise Design and Engineering Working Conference, November 28 – December 1, 2023, Vienna, Austria
*Corresponding author.
lukas.pfahlsberger@hu-berlin.de (L. Pfahlsberger); christoffer.rubensson@hu-berlin.de (C. Rubensson); steven.knoblich@hu-berlin.de (S. Knoblich); maxim.vidgof@wu.ac.at (M. Vidgof); jan.mendling@hu-berlin.de (J. Mendling)
https://www.informatik.hu-berlin.de/en/forschung-en/gebiete/promis-en/team/pfahlsbl (L. Pfahlsberger); http://hu.berlin/rubensson (C. Rubensson); https://nm.wu.ac.at/nm/vidgof (M. Vidgof); http://hu.berlin/mendling (J. Mendling)
0000-0002-1367-9441 (L. Pfahlsberger); 0009-0004-4940-5866 (C. Rubensson); 0009-0002-8509-7042 (S. Knoblich); 0000-0003-2394-2247 (M. Vidgof); 0000-0002-7260-524X (J. Mendling)

In this paper, we propose our vision for multi-perspective path semantics. To this end, we integrate causal process knowledge [2] that allows differentiating between eight different path semantics. Furthermore, we abstract the individual path semantics into distinguishable pattern types that are categorized into allowed and prohibited behavior. This allows linking pattern types directly to generic use cases for process analysis. Thereby, analysts can identify and discover interesting behavior that indicates problems in visual process representations more precisely.

We contribute to the field of visual analytics for process mining by proposing multi-perspective path semantics based on causal process knowledge. Our approach improves the precision and speed of the process analysis due to a linkage between the meaning of the paths and suitable use cases. Previous studies have presented visualization frameworks for process mining [3] that investigate the effectiveness of different ways of representing process mining outcomes. However, these works often neglect the aspect of path semantics and instead focus on alternative visual forms of aggregating, clustering, or sorting the process data. We further contribute to the integration of prior domain-specific knowledge into existing process discovery techniques [4, 5, 6, 7]. We thereby show that prior knowledge can not only help to improve the structural aspects of the model (e.g., by reducing complexity) but also the visual representation of its elements, such as paths.

The remainder of the paper is structured as follows. Section 2 presents the theoretical background focusing on visual analytics, graph types in process mining, with a focus on arc semantics, and causal process knowledge. Section 3 introduces our vision for multi-perspective path semantics. Section 4 derives pattern types based on distinct combinations of path semantics on a process instance level. Section 5 links the pattern types to use cases, thus evaluating our vision for multi-perspective path semantics based on a real-world order-to-cash process. Section 6 points out implications for future research and limitations. Finally, Section 7 concludes this paper.

2. Background

In this section, we discuss three distinct areas of research. First, we briefly introduce the topic of visualization and how it is utilized for analytical purposes. Second, we introduce previous works on the semantics of arcs in process models. Finally, we define causal process knowledge and explore its applications in process mining.

2.1. Visualization & visual analytics

Visualization can be seen as the process of converting data into graphical representations [8, p. 3], enabling the derivation of insights otherwise difficult to discern from raw data sets. Visual analytics is a multidisciplinary field that employs visualization techniques for graphically representing knowledge and enhancing analytical reasoning [9, p. 4]. Many visualization techniques are available to accomplish this objective (cf., [10, 11]).

Visual analytics relies on active interaction between users and data [9, p. 4], making human judgment a crucial component. Munzner's *nested model* [12], a design and validation framework for visualizations, emphasizes a human-centered approach with a four-layer design process. The

framework takes the domain problem and its intended user as a starting point. This is followed by translating the problem into a computer science context, designing the visualization, and finally creating its rendering mechanism. McKenna et al. [13] extend the nested model [12] with the overlapping activities *understand*, *ideate*, *make*, and *deploy* that further emphasize the user-centric motivation and their design outcomes for each step in the process. In another framework, Moere and Purchase [14] define roles with domain-specific needs for visualizations, which could facilitate the quality of the design solutions when met. These roles comprise the *visualization studies* (researchers), with the aim for utility and soundness; the *visualization practice* (businesses), with the need for market-oriented solutions; and the *visualization exploration* (artists), who strive to create visually appealing yet workable designs [14, pp. 366-368]. Lastly, Moody [15] developed the *Physics of Notations*, a theory comprising a set of design principles to support the creation and validation of cognitively effective visualizations in software engineering. An example principle is the *Principle of Semiotic Clarity* that ascribes designers to ensure a one-to-one relationship between graphical symbols and the semantic construct they represent [15, pp. 762-763]. Failure to adhere to this principle may result in ineffective visualizations, such as a *symbol deficit* whenever a semantic is not represented by any symbol or a *symbol redundancy* whenever multiple symbols refer to the same semantic [15, pp. 762-763].

The process of visualizing the data involves a range of dimensions to consider (cf., [15]). While we only provide a few examples, one aspect involves using geometrical objects to depict data, such as glyphs in different shapes [8]. A well-known depiction of data in statistics is the boxplot (e.g., [16, pp. 45-46]), in which numerical data is grouped into a single box with extending lines (whiskers) to indicate, i.a., data variability. In process science, a standard for visualizing processes is the *Business Process Model and Notation* (BPMN) standard[1], which, i.a., uses boxes, rhombuses, and arrows to, respectively, depict *activities*, *gateways* (decision points), and *sequence flow* between activities. Another aspect to consider is *color*, or color mapping, which can be used to map data to certain attributes [8, p.5]. Despite appearing trivial, color in visualization is a highly complex topic as it is closely linked to human perception through various channels, such as color properties (e.g., hue and saturation), color combinations, and geometrical patterns (e.g., [17]). Brewer [18] makes an important contribution that provides insights and guidelines on color mapping based on data types and human perception, further demonstrating the topic's complexity.

2.2. Arc semantics in process models

Some of the existing process modeling languages already provide arcs with different semantics. However, the purpose and concepts behind such differentiation is not the same across languages. On the other hand, some languages do not distinguish between different arc semantics. In *directly-follows graphs* [1], for instance, there is only a single type of arc connecting two activities. Its arc semantics simply considers observed events that follow the path between the two activities. These arcs can store additional information (e.g., the number of instances following the arc) or durations of transitions. In BPMN, there are two kinds of arcs, namely, *control flow* and *message flow*. The semantics of the former is that a process instance transitions

[1] https://www.bpmn.org (Last accessed: 2023-12-10)

from one activity to another. The semantics of the message flow is that a message from an external pool (e.g., sent from an external party such as an organization) is received by an activity or event in another pool or vice versa [19].

To the best of our knowledge, *Petri nets* have the largest variety of arc semantics. While the primary purpose of arcs in Petri nets is to represent the movement of the tokens between *places* and *transitions*, the exact semantics may differ. First, there are differences in terms of whether and how many tokens are moved. Traditionally, one token moves along the arc when it is consumed or produced by a transition. However, there are also *read arcs* [20] requiring tokens to be present in a place for a connected transition to fire but not consuming or producing the tokens. There are *inhibitor arcs* [21] that, on the contrary, prevent an otherwise enabled transition to fire if there are tokens in a specified place. *Weighted arcs* [22] specify exactly how many tokens are consumed or produced by a transaction, allowing more than one token to be moved along an arc. Finally, *reset arcs* [23] remove all tokens from respective places when a transition fires.

Furthermore, the type of tokens being moved can also differ. *Colored Petri nets* [24] allow distinguishing between different types of objects or object instances by assigning colors to tokens. For the places, one can then define different capacities for tokens of different colors. The firing semantics of the transitions can also depend on the color. Finally, the arcs ultimately specify the colors of tokens to be consumed or produced.

2.3. Causal process knowledge in process mining

As a forefather of the philosophy of *causation*, Hume [25] noted that all knowledge comes from experience and that it is based on associations between perceived events. Waldmann [26] adopted this idea in his work on *knowledge-based causal induction*, indicating causal directionality as the fundamental factor for determining how statistical correlations are understood. The term *causation* can be further differentiated by Pearl's [27] three-level causal hierarchy highlighting the role of causal knowledge in helping to associate, intervene, or counterargue.

Regarding causal knowledge concerning business processes, experts with years of acquired domain-specific experience represent a valuable resource for process improvement. Experience provides process experts with a precise understanding of causal relationships between individual activities of business processes. For instance, a process owner of an order-to-cash process might readily understand that a customer order eventually leads to an invoice being created. Intuitively, it is clear to the process owner that, oppositely, an invoice followed by the customer order would contradict the causal logic of the process [2].

In the process mining research field, most discovery algorithms do not leverage causal process knowledge [2, 4]. Instead, they consider data as the "single source of truths" to behaviors while overlooking domain-specific reasons. In an experimental setting, Rembert et al. [4] develop and test a process discovery algorithm that integrates prior knowledge. The results indicate that prior knowledge increases the robustness against noise, subsequently reducing the likelihood of measurement and ordering errors, particularly for processes with a higher degree of infrequent behavior. Similarly, Diamantini et al. [5] exploit knowledge in complex domains with highly variable processes as a means to repair event logs and produce more realistic models. Waibel et al. [2] use a causal template that helps process analysts integrate a causal order into discovering

the process structures with a focus on control-flow. Compared to approaches that do not integrate domain knowledge, the approach by [2] generates much simpler models with higher conformance to the defined causality by reducing the number of self-loops and spurious arcs. Lu et al. [6] propose a semi-automated approach to detecting log patterns in process discovery, using human reasoning to evaluate, modify, and extend pattern types.

3. Multi-perspective path semantics

In this section, we present our vision of multi-perspective path semantics. To this end, we develop eight path semantics through visual characteristics related to shape and color. The semantics differ based on whether the path indicates desired or undesired behavior, observed or unobserved behavior, and its flow direction.

3.1. Conformance path

The semantics of the *conformance path* relates to a combination of desired and observed behavior. On the one hand, desired behavior connects to the causal process knowledge, which the analyst predefines as a working hypothesis. By that, analysts presuppose, based on their own expertise, that within the very specific context, the process is supposed to flow through this particular path. It is implicitly assumed that process behavior not flowing through this path is understood as a deviation or at least as an unexpected behavior. On the other hand, the conformance path additionally incorporates the observed behavior that is recorded in the data source. Hence, the semantics of the conformance path can be considered both an *actually* observed behavior in the source data and a desired behavior, as intended by the analyst.

The visual representation takes the form of an arrow with a solid gray line connected to a filled arrowhead. The structure of this path runs angularly and has no curvatures. The intention behind the visualization is to convey the impression of desired behavior that is not explicitly prominent.

3.2. Hypothetical path

The semantics of the *hypothetical path* relates to an unobserved yet desired behavior. This means that the causal process knowledge allows the process to flow through this particular path with no record in the data confirming this behavior. If a hypothetical path occurs, this always implies that there is at least one other path option over which the process can flow as well. For instance, this can be attributable to the causal process knowledge allowing for the parallel execution of two activities with a time offset or a passage in the process that allows for an arbitrary choice of follow-up options.

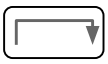
The visual representation takes the form of an arrow with a gray dashed line connected to a filled arrowhead with an identical angular course. The visualization is supposed to convey the indefinite characteristics of a non-conforming behavior.

3.3. Omitted path

The semantics of the *omitted path* also relates to unobserved yet desired behavior. As opposed to the hypothetical path, here, the intended sequence flow is considered mandatory, but with no data recorded that confirms its execution. If an omitted path occurs, it can be concluded that certain process activities have been skipped unintentionally or that the order of activities has been reversed.

The visual representation takes the form of an arrow with a gray dashed line connected to an unfilled (empty) arrowhead. It also runs in an angular course. The unfilled arrowhead should give the impression that the path is mandatory, thus its exaggerated appearance.

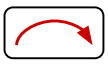
3.4. Allowed shortcut path



The semantics of the *allowed shortcut path* relates to desired and observed behavior. Therefore, the causal process knowledge indicates that the process is allowed to skip one or more process activities without following up on them at later points, and the data recorded indicates that this, in fact, happened. If this path appears in the model, a hypothetical path can be linked to it because a circuitous route via the activities skipped, in reality, would also have been possible.

The visual representation takes the form of an arrow with a solid gray line connected to a filled arrowhead. The structural appearance, again, follows an angular course with the intention to convey expected and desired behavior. However, this path can often be recognized as flowing in parallel to the direction of other *conformance paths*.

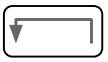
3.5. Prohibited shortcut path



The semantics of the *prohibited shortcut path* relates to undesired yet observed behavior. In this case, the causal process knowledge explicitly does not allow the process to jump over a specific activity but is recorded in the data. Here, at least one omitted path can be linked to the prohibited shortcut path because other activities not intended to be executed were left out and not followed up on.

The visual representation takes the form of an arrow with a solid red line connected to a filled arrowhead. In this case, as opposed to the allowed shortcut, it has a curvilinear course. Here, contrast is to be conveyed in relation to the allowed shortcut path, which indicates an undesirable behavior through the round and less structured-appearing course.

3.6. Allowed backjump path



The semantics of the *allowed backjump path* relates to observed and desired behavior. The causal process knowledge indicates that the process can jump back to already executed process activities. Once an allowed backjump occurs, there are multiple follow-up options leading to an increased level of complexity.

The visual representation takes the form of an arrow with a solid gray line connected to a filled arrowhead. Thereby, the path runs in an angular course in the opposite direction of other

conformance paths. Even though backjumps in processes may be negatively conjugated, this path semantics emphasizes the acceptance to repeat an activity already executed before.

3.7. Prohibited backjump path

The semantics of the *prohibited backjump path* relates to observed yet undesired behavior. In this case, the causal process knowledge restricts the process from not returning to a previously performed activity despite the recorded data indicating otherwise. With these paths, a large variety of follow-up options becomes possible, which usually leads to higher degrees of complexity.

The visual representation takes the form of an arrow with a solid red line connected to a filled arrowhead with a curvilinear course. In most cases, this path runs in the opposite direction of other *conformance paths*, indicating an undesired behavior.

3.8. Skip-reverse path

The semantics of the *skip-reverse path* relates to observed yet undesired behavior, which inevitably occurs in combination. This means that the recorded behavior indicates that the process activities were executed in an order that the causal process knowledge forbids. This triggers at least one path that skips a considered follow-up activity and at least one reverse path that continues where another activity was left out. However, even though all intended process activities were executed, the order of activity execution was incorrect. If this path is shown in the model, an omitted path can always be linked to a pair of reverse and skipped paths.

The visual representation takes the form of an arrow with a solid red line connected to a filled arrowhead with a curvilinear course. Together with the inseparably connected path to the follow-up activity and the omitted path between the activities in the original order, the path semantics intends to create a complex-appearing and slightly chaotic impression that conveys that something is not going as desired.

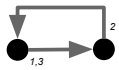
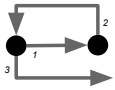
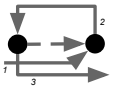
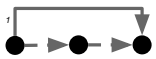
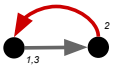
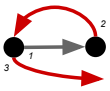
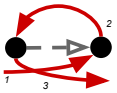
4. Pattern types

This section briefly demonstrates how the context, meaning various arrangements of different path semantics, may emphasize an underlying behavior that is otherwise hard to distinguish when examining one single path semantics in isolation. We refer to these *path semantics ensembles* as *patterns* or pattern types on a process instance level. An example of this is the allowed backjump path, which could denote a wrong-order execution of events if it is proceeded by a hypothetical path rather than when proceeded by a conformance path.

We identify eight possible patterns, all listed in Table 1. The patterns are divided into two categories, *allowed* and *prohibited*, whether they include allowed or prohibited path semantics. Each pattern is given a name for easy recognition of an underlying behavior. We also exemplify the patterns from the perspective of an order-to-cash process.

To further enhance understandability, we define a pattern as a sequence of directly follows relations $\langle a_1 \rightarrow a_2 ... \rightarrow a_n \rangle$ that is part of a process instance, where $a_1 ... a_n$ are activities of the

Table 1
An overview of some possible pattern types that result from a unique set of path semantics. The numbers indicate the execution order, with "1" denoting the first execution.

Path combinations	*Backjump Conformance*	*Backjump/Shortcut Conformance*	*Backjump Hypothetical/Omitted*	*Shortcut Hypothetical/Omitted*
Allowed	 Refinement	 Adjustment	 Reorder	 Simplification
Prohibited	 Rework	 Correction	 Disarray	 Negligence

same sequence with different activity types so that $a_1 \neq ... \neq a_n$ holds. The timestamps of two activities in a directly follows relation $a_1 \rightarrow a_2$ must be different so that $t(a_1) < t(a_2)$ holds. If an activity (type) a_1 is executed twice in the instance, the timestamps must be different so that $t(a_1) < t(a_1')$ holds. The arrow symbol $\rightarrow$ denotes a directly follows relations between two activities a_1 and a_2 in a pattern, which is further assigned a letter to indicate its path semantics as either a backjump path $\rightarrow_B$, a shortcut path $\rightarrow_S$, a conformance path $\rightarrow_C$, a hypothetical path $\rightarrow_H$, or an omitted path $\rightarrow_O$. A crossed-out arrow $\nrightarrow$ emphasizes a prohibited path. Note that we do not differentiate between designed and executed sequences since the path semantics imply it.

4.1. Allowed pattern types

The allowed pattern types comprise path combinations, which are allowed by design. These include: *refinement*, *adjustment*, *reorder*, and *simplification*.

Refinement is characterized by a backjump path that follows and is followed by two identical conformance paths leading to the emerging sequence $\langle a_1 \rightarrow_C a_2 \rightarrow_B a_1 \rightarrow_C a_2 \rangle$. Since the same directly follows relations $a_1 \rightarrow_C a_2$ is executed twice, this pattern indicates a revising behavior (e.g., when a customer proofreads an order detail before purchase).

Adjustment is characterized by a conformance path followed by a backjump path, which sequentially is followed by a shortcut path, leading to the emerging sequence: $\langle a_1 \rightarrow_C a_2 \rightarrow_B a_1 \rightarrow_S a_3 \rangle$. Here, only the first activity a_1 is executed twice (cf., *refinement*), thus indicating a re-routing of an intended activity sequence (e.g., when a customer cancels a requested credit card payment and, instead, decides to pay in installments).

Reorder is characterized by a backjump path that follows a hypothetical path, which sequentially is followed by a shortcut path leading to the emerging sequence: $\langle a_1 \rightarrow_H a_2 \rightarrow_B a_1 \rightarrow_S a_3 \rangle$. This means that the intended activity sequence is executed in reverse (e.g., when an order foresees a purchase-for-delivery procedure when, in fact, customers purchase items (through bill) after delivery).

Simplification is characterized by a shortcut path that skips a sequence of hypothetical paths, such that the following pattern occurs: $\langle a_1(\rightarrow_H \ldots \rightarrow_H a_n, \rightarrow_S a_{n+1})\rangle$. This pattern suggests redundancies in the process design (e.g., when customers use an autofill function to fill out their demographic details before purchase).

4.2. Prohibited pattern types

The prohibited pattern types are analogous to the allowed patterns but contain at least one prohibited path, thus indicating violations of an intended process design. These include: *rework*, *correction*, *disarray*, and *negligence*.

Rework is the analog to *refinement* yet with prohibited paths, such that the following pattern emerges: $\langle a_1 \rightarrow_C a_2 \nrightarrow_B a_1 \rightarrow_C a_2\rangle$. Here, a refining behavior is instead a source of frustration or an unnecessary emendation (e.g., when a customer has to re-purchase an order after discovering the purchase of the wrong items).

Correction is the analog to *adjustment* yet with prohibited paths, such that the following pattern emerges: $\langle a_1 \rightarrow_C a_2 \nrightarrow_B a_1 \nrightarrow_S a_3\rangle$. In comparison, whereas an *adjustment* may improve a process towards a better outcome, in *correction,* an avoidable mistake is adjusted to prevent harm (e.g., when a customer must be contacted after they were able to purchase an order with a suspended credit card successfully).

Disarray is the analog to *reorder* yet with prohibited paths, such that the following pattern emerges: $\langle a_1 \rightarrow_O a_2 \nrightarrow_B a_1 \nrightarrow_S a_3\rangle$. Here, a rearrangement of (strict) protocol procedure is executed (e.g., when an order is marked as successful before verification).

Negligence is the analog to *simplification* yet with prohibited paths, such that the following pattern emerges: $\langle a_1(\rightarrow_O \ldots \rightarrow_O a_n, \nrightarrow_S a_{n+1})\rangle$. Here, the complete skipping of an intended activity sequence is considered wrong rather than as an improvement (e.g., when a customer is warranted a replacement item after a filed complaint without the warranty not being properly inspected by the company).

5. Use cases

Allowed patterns reflect expected behavior and give insights into how well a process is adopted. Prohibited patterns are more complex. In this section, we focus on prohibited pattern types as we expect more business value from their analysis. Therefore, we articulate three assumptions and review the pattern according to four performance dimensions relevant to business processes. In the second part of this section, we examine what a technical solution can look like and explain the business impacts that can be derived from a specific process instance of an order-to-cash process. We then further apply heuristics to improve the process.

5.1. Assumptions

Previous research addresses the speed of technical development and its adoption in business, which leads to the clear call to action of transferring new developments in real-life use cases [28]. Some concepts are highly adopted in business, such as the *Balanced Scorecard* with its four

Table 2
Impact of prohibited pattern types on the performance perspectives time, cost, and quality

		Performance Perspective		
		Time	*Cost*	*Quality*
Pattern	*Rework*	↓↓	↓↓	↑
	Correction	↓	○	↑
	Disarray	○	○	↓
	Negligence	↑	↑	↓↓

Legend: ↑↑ high positive impact, ↑ medium positive impact, ○ no impact, ↓ medium negative impact, ↓↓ high negative impact

perspectives financial, customer, learning, and growth as well as internal business process [29][2]. The latter, namely the internal business process perspective, can be measured by the four performance dimensions of the *devil's quadrangle*, namely, (1) time, (2) cost, (3) quality, and (4) flexibility [19]. We apply this to emphasize that the improvement of one or multiple perspectives results in less performance of at least one other perspective [19]. For this paper, we review only the prohibited pattern types of Section 4 and the impact on the four performance dimensions with three assumptions, knowing that under realistic conditions, there are cases that will not fulfill them. We use the following assumption:

1. With increasing complexity due to path variants and activity numbers, the average process instance duration increases
2. Every activity has a cost, mainly labor costs, resulting in a negative financial impact per executed activity.
3. Every activity adds value and, therefore, enhances the quality of the process outcome, resulting in better quality the more (planned) activities are performed.

Based on these assumptions, the impact on process performance is summarized in Table 2. As no impact on flexibility is identified, we do not address this perspective.

- The *rework* pattern repeats two events and adds two connections, resulting in a high negative impact on cost. The quality is benefiting from the rework, as it repairs an error.
- The *correction* pattern repeats one event and adds one additional connection, resulting in a medium negative impact on time and costs. The quality is benefiting as an unexpected result is prohibited.
- The *disarray* pattern impacts time and cost under respecting the assumption, but a medium negative impact on quality as the sequence of events is not followed.
- The *negligence* pattern is skipping one event and having one connection less, resulting in a positive effect on costs and time. On the other hand, as an event is skipped, a high negative effect on quality is expected.

[2] https://www.bain.com/insights/management-tools-and-trends-2023/ (Last accessed: 2023-12-10)

5.2. Example

To illustrate the added value of multi-perspective path semantics for business process analysis, we showcase its application using a real-world example of an order-to-cash process observed at a German mid-sized company. To visualize this process, we use a tool of Noreja[3]. This order-to-cash process starts with placing a customer order, which is followed by the preparation and shipping of digital or physical goods or services and ends with financial processing, including the posting of an invoice and receiving of cash. This first extract of a process instance in Figure 1 represents the pattern type disarray[4]. After the event *Create Delivery Note* (left of Figure 1), the process continues with *Receive Payment*, skipping the actually desired follow-up activity *Post Invoice*. This leads to an undesired order of the events *Post Invoice* and *Receive Payment* that contradicts the causal process knowledge. Due to the particular semantics of the paths and their highlighting in red, process analysts can now directly identify this pattern in order to derive actions. The omitted paths indicate the desired process relations. In this case, the pattern indicates that the organization takes a financial risk, as matching the payment against the actual invoice is not secured. In this example, the payment terms for each customer order are highly different and optimized by the sales department in terms of discounts, overdue fines, etc. The lack of invoice-payment matching, therefore, causes significant problems. When receiving the payment before the invoice, the potential of overpayment or underpayment is given, resulting in lower customer satisfaction and more inaccurate financial planning.

Over the last decades, several redesign methodologies have been developed to improve process performance, including redesign heuristics [19]. One of these heuristics is the *case-base work* that removes the processing of cases in batches or at specific points in time (e.g., every Monday). With handling individual cases, the time between two events can be shortened. Applying this heuristic on the event post invoice can significantly reduce the time from creating the delivery note to the invoice posting and reduce the risk of receiving payment with an invoice reference. Applying this heuristic will, under assumption 2, result in additional costs, as the invoice posting will happen more frequently. Another heuristic that can solve the downside of the case-based work is the *activity automation*. Automating the posting of the invoice and executing this event directly after the creation of the delivery note will reduce the time between these events and the costs for the posting.

6. Limitations and future research

This paper presents multi-path semantics for process mining, pattern types, and possible applications. This research stream is novel and promising, yet covering it entirely in one study is infeasible. We acknowledge the limitations of our paper and use them to outline directions for future research.

First, we must note the limitations of the scope of our paper. We explicitly exclude so-called *change activities*[5]. In addition, as the primary goal of this paper is to highlight the necessity for

[3] https://noreja.com (Last accessed: 2023-12-10)

[4] The blue symbols of the events themselves are not part of our proposed concept and therefore not explained.

[5] *Change activities* cannot be sorted into the causal logic of a process, as they may appear randomly due to unplanned occurrences. In an order-to-cash process, this takes the form of cancellations or price/quantity changes.

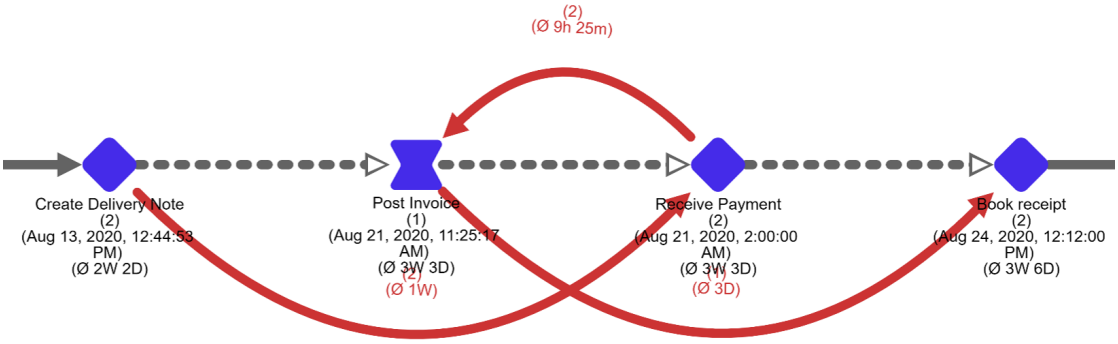


Figure 1: Visual representation of the pattern type *disarray* in a single process instance of an order-to-cash process

multi-path semantics rather than to provide an exhaustive list, we note that other semantics, including domain-specific ones, may be defined by future extensions. The same applies to the patterns presented in this paper. While we identified some critical patterns using the proposed multi-path semantics, additional patterns might also be observed, especially if new path semantics are added.

Second, it must be noted that in this paper, we are only considering possible path semantics for single cases. However, new challenges will arise as we try to raise the level of abstraction (e.g., to a variant or even process level). Aggregating cases with paths of different semantics between the same activities or aggregating path patterns is a non-trivial task. For instance, if, in one case, activities a_1 and a_2 are connected via a conforming path. In another case, the same activities are only connected via an omitted path. Then, it remains unclear which semantics (and which visual representation) should be chosen when aggregating on a process level.

Third, while we do provide some visual descriptions of the paths with different semantics, it must be noted that these descriptions should be treated as preliminary proposals for visualization rather than fixed recommendations or guidelines. We explicitly leave the specific visualization (e.g., a more differentiated coloring of paths) out of the scope of this paper. We also note that further visual additions can be made (e.g., additional icons near the arc ends) for improved visual differentiation and reduced cognitive load. Ultimately, we highlight that a user evaluation is required to validate the visualization approach.

In future work, we plan to tackle the identified limitations. First, empirical research is needed to evaluate both the relevance of the proposed path semantics in practice as well as the suitability of various visualization approaches. Second, further path semantics and pattern types can be obtained using both empirical and explorative studies. Finally, the impact of the observed paths on flexibility – the fourth dimension of process performance – is yet to be studied.

7. Conclusion

In this paper, we envisioned a foundation for multi-perspective path semantics in process mining. First, we presented eight distinct path semantics that can be derived by integrating causal process knowledge. Second, we defined pattern types as unique sets of path semantics on a process instance level to provide additional meaning about underlying behaviors. We demonstrated the benefits of our approach by exemplifying the semantics in a use-case scenario, examining various pattern types according to their business values using the *devil's quadrangle*. In addition, we gave some example visualizations from an existing process mining system. Our work facilitates the interpretability and applicability of process mining outcomes by providing a more fine-grained view of path semantics. By linking causal process knowledge to visual elements, we further contribute by extending the graphical capabilities of process models. In combination, both aspects facilitate the fast and purposeful acquisition of process-related insights for analysts. In this way, our objective is to inspire future research to challenge the still prevailing representational bias [30] in process mining, which oftentimes distorts the true nature of the underlying process.

Acknowledgments

This work was supported by the Einstein Foundation Berlin [grant number EPP-2019-524, 2022] and Deutsche Forschungsgemeinschaft [grant number ME 3711/2-1].

References

[1] W. M. P. van der Aalst, Process Mining - Data Science in Action, Second Edition, Springer, 2016. doi:10.1007/978-3-662-49851-4.

[2] P. Waibel, L. Pfahlsberger, K. Revoredo, J. Mendling, Causal process mining from relational databases with domain knowledge, CoRR abs/2202.08314 (2022). URL: https://arxiv.org/abs/2202.08314.

[3] A. Yeshchenko, J. Mendling, A survey of approaches for event sequence analysis and visualization, Information Systems 120 (2024) 102283. doi:10.1016/j.is.2023.102283.

[4] A. J. Rembert, A. Omokpo, P. Mazzoleni, R. Goodwin, Process discovery using prior knowledge, in: S. Basu, C. Pautasso, L. Zhang, X. Fu (Eds.), Service-Oriented Computing - 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings, volume 8274 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 328–342. doi:10.1007/978-3-642-45005-1_23.

[5] C. Diamantini, L. Genga, D. Potena, W. M. P. van der Aalst, Building instance graphs for highly variable processes, Expert Syst. Appl. 59 (2016) 101–118.

[6] X. Lu, D. Fahland, R. Andrews, S. Suriadi, M. T. Wynn, A. H. M. ter Hofstede, W. M. P. van der Aalst, Semi-supervised log pattern detection and exploration using event concurrence and contextual information, in: OTM Conferences (1), volume 10573 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 154–174.

[7] W. Song, H. Jacobsen, C. Ye, X. Ma, Process discovery from dependence-complete event logs, IEEE Trans. Serv. Comput. 9 (2016) 714–727. doi:10.1109/TSC.2015.2426181.
[8] W. J. Schroeder, K. M. Martin, 1 - Overview of visualization, in: C. D. Hansen, C. R. Johnson (Eds.), Visualization Handbook, Butterworth-Heinemann, Burlington, 2005, pp. 3–35. doi:10.1016/B978-012387582-2/50003-4.
[9] J. J. Thomas, K. A. Cook (Eds.), Illuminating the Path: The Research and Development Agenda for Visual Analytics, National Visualization and Analytics Center, 2005. ISBN: 0-7695-2323-4.
[10] M. C. F. de Oliveira, H. Levkowitz, From visual data exploration to visual data mining: A survey, IEEE Trans. Vis. Comput. Graph. 9 (2003) 378–394. doi:10.1109/TVCG.2003.1207445.
[11] W. Cui, Visual analytics: A comprehensive overview, IEEE Access 7 (2019) 81555–81573. doi:10.1109/ACCESS.2019.2923736.
[12] T. Munzner, A nested model for visualization design and validation, IEEE Trans. Vis. Comput. Graph. 15 (2009) 921–928. doi:10.1109/TVCG.2009.111.
[13] S. McKenna, D. Mazur, J. Agutter, M. D. Meyer, Design activity framework for visualization design, IEEE Trans. Vis. Comput. Graph. 20 (2014) 2191–2200. doi:10.1109/TVCG.2014.2346331.
[14] A. V. Moere, H. C. Purchase, On the role of design in information visualization, Inf. Vis. 10 (2011) 356–371. doi:10.1177/1473871611415996.
[15] D. L. Moody, The “physics" of notations: Toward a scientific basis for constructing visual notations in software engineering, IEEE Trans. Software Eng. 35 (2009) 756–779. doi:10.1109/TSE.2009.67.
[16] K. Backhaus, B. Erichson, S. Gensler, R. Weiber, T. Weiber, Multivariate Analysis: An Application-Oriented Introduction, 1 ed., Springer Gabler Wiesbaden, Wiesbaden, 2021.
[17] L. Bartram, A. Patra, M. C. Stone, Affective color in visualization, in: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017, ACM, 2017, pp. 1364–1374. doi:10.1145/3025453.3026041.
[18] C. A. Brewer, Chapter 7 - Color use guidelines for mapping and visualization, in: Modern Cartography Series, volume 2, Elsevier, 1994, pp. 123–147. doi:10.1016/B978-0-08-042415-6.50014-4.
[19] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Springer, 2013.
[20] W. Vogler, A. L. Semenov, A. Yakovlev, Unfolding and finite prefix for nets with read arcs, in: D. Sangiorgi, R. de Simone (Eds.), CONCUR ’98: Concurrency Theory, 9th International Conference, Nice, France, September 8-11, 1998, Proceedings, volume 1466 of *Lecture Notes in Computer Science*, Springer, 1998, pp. 501–516. doi:10.1007/BFb0055644.
[21] M. H. T. Hack, Petri net language, Technical Report, USA, 1976.
[22] W. Reisig, Petri Nets: An Introduction, volume 4 of *EATCS Monographs on Theoretical Computer Science*, Springer, 1985. doi:10.1007/978-3-642-69968-9.
[23] T. Araki, T. Kasami, Some decision problems related to the reachability problem for petri nets, Theor. Comput. Sci. 3 (1976) 85–104. doi:10.1016/0304-3975(76)90067-0.
[24] K. Jensen, Coloured petri nets and the invariant-method, Theor. Comput. Sci. 14 (1981) 317–336. doi:10.1016/0304-3975(81)90049-9.

[25] D. Hume, An Enquiry Concerning Human Understanding, Hackett Publishing Company, Indianapolis, IN, 1977. Original work published 1748.
[26] M. R. Waldmann, Knowledge-based causal induction, in: Psychology of Learning and Motivation, volume 34, Elsevier, 1996, pp. 47–88.
[27] J. Pearl, The seven tools of causal inference, with reflections on machine learning, Commun. ACM 62 (2019) 54–60. doi:10.1145/3241036.
[28] J. vom Brocke, M. Jans, J. Mendling, H. A. Reijers, Call for papers, Issue 5/2021, Business & Information Systems Engineering 62 (2020) 185–187.
[29] R. S. Kaplan, D. P. Norton, The Balanced Scorecard: Translating strategy into action, Harvard Business School Press, Brighton, MA 02135, 1996.
[30] W. M. P. van der Aalst, J. C. A. M. Buijs, B. F. van Dongen, Towards improving the representational bias of process mining, in: K. Aberer, E. Damiani, T. S. Dillon (Eds.), Data-Driven Process Discovery and Analysis - First International Symposium, SIMPDA 2011, Campione d'Italia, Italy, June 29 - July 1, 2011, Revised Selected Papers, volume 116 of *Lecture Notes in Business Information Processing*, Springer, 2012, pp. 39–54. doi:10.1007/978-3-642-34044-4_3.