Notes of a Process Scientist: Process Discovery

Artem Polyvyanyy

The University of Melbourne, Australia
artem.polyvyanyy@unimelb.edu.au

Abstract. Process discovery uses event data generated by information systems to construct process models that describe system behavior. This paper revisits the problem from a conceptual standpoint and clarifies its core elements, including the nature of the input event data, the intended characteristics and purposes of the resulting models, the variants of the discovery problem, the key challenges associated with solving it, and the principles that guide the evaluation of the quality of discovered models. We further position process discovery in relation to related problems in adjacent research areas and identify open issues and research opportunities. Through this analysis, the paper aims to provide a solid reference point for future methodological and tool developments in process discovery.

Keywords: Process mining, process discovery, variants, challenges, related problems, quality, future directions

1 Introduction

Process mining is a discipline that bridges data science and process science. It studies how processes executed by systems in the real world can be discovered, monitored, and improved by analyzing event data recorded during their execution [2]. At its core, process mining seeks to answer how a system actually behaves, including which activities occur, how often and in what order they are executed, under which conditions and by whom they are performed, and what outcomes they produce. Over the past two decades, process mining has evolved into a mature research field, influencing both academia and industry, with techniques now applied across domains such as business operations, healthcare, manufacturing, and software engineering. Among the key problems in process mining, process discovery stands as the foundational one. Given event data, *process discovery* aims to construct a process model that meaningfully represents the behavior of the system that generated the data [8].

This paper is the first in a series of opinion pieces under the heading *Notes of a Process Scientist*, in which we examine foundational problems in process science. Here, we focus on the problem of process discovery as it should be understood in light of its purpose and scientific roots. The goal is to contribute to the discussion about what process discovery aims to achieve, what its results should explain, and how its success should be evaluated. In this sense, the paper is not a survey of existing methods, but a conceptual reflection on the essence and future direction of process discovery.

The remainder of the paper proceeds as follows. The next section introduces the fundamental concepts underlying process discovery and provides a formal definition

of the problem. Section 3 then outlines important variants of the conventional problem formulation that differ in the behavioral aspects of the system they capture in discovered models. Section 4 discusses the main challenges associated with discovering process models from event data. Subsequently, Section 5 reviews related problems explored in adjacent research areas. Finally, Section 6 examines how the quality of discovered models can be evaluated, before Section 7 concludes the paper.

2 Process Discovery

This section presents key concepts relevant to process discovery, namely systems (Section 2.1), event data (Section 2.2), processes (Section 2.3), and process models (Section 2.4), and gives a formal definition of the process discovery problem (Section 2.5).

2.1 Systems

A dynamical system, or simply a *system*, consists of interacting components whose behavior changes over time. The behavior of a system can be described by its states and by rules that determine how it transitions between states. A state is a collection of attributes and their values that characterize the condition of the system at a given point in time. A state can be a global snapshot that abstracts away the internal complexity of individual components and captures only the information needed to describe the system as a whole. Alternatively, the global state of a system can be understood as the aggregation of the local states of its interacting components [35].

The state of a continuous-time system evolves smoothly at every instant, whereas in a discrete-time system, state updates occur at distinct time steps. Discrete systems are particularly relevant in the context of information systems. An information system consists of interacting software components that perform activities by consuming, producing, and manipulating information. Its states change at identifiable, discrete events.

2.2 Event Data

In the context of systems, *data* refers to the measured and recorded values that describe the observed states of a system, the inputs it consumes, the outputs it produces, the activities that trigger state changes, or the characteristics of the environment in which the system is embedded. To associate each measurement with a specific moment in the life of the system, it is timestamped and represented as an *event*. Whereas states and state-transition rules define the principles that govern how the system evolves, data form the observed and recorded traces of that evolution. In this way, data provide empirical evidence of how the system behaves in the real world.

Let $\mathcal{A}$ be a universe of *attributes* and let $\mathcal{V}$ be a universe of *values*, such that $timestamp \in \mathcal{A}$ is a special timestamp attribute. Then, an *event* e is a partial function $e : \mathcal{A} \nrightarrow \mathcal{V}$ that assigns a value to the timestamp attribute; that is, $timestamp \in dom(e)$ and $e(timestamp) \in \mathcal{T}$, where $\mathcal{T} \subset \mathcal{V}$ is a set of timestamps totally ordered by $\leq$. The value $e(timestamp)$ specifies the point in time at which event e was observed and/or recorded. Finally, *event data* is a collection of events.

Table 1 presents an example of event data comprising twenty events. Each row in the table defines a single event with three attributes: *case*, *activity*, and *timestamp*. For instance, the last row defines the event with the values of case, activity, and timestamp attributes set to 3, d, and 20, respectively. This event was triggered at timestamp 20 by activity d executed as part of process case 3.

2.3 Processes

A *process instance* is a sequence or ordering of activities that a system performs, or is capable of performing, in pursuit of achieving a specific goal. Each activity transforms the system by inducing a state transition, and the entire process instance leads the system from an initial state to a final state. A *process* is a collection of such instances that (aim to) accomplish the same goal. For example, an information system may automate the process of handling customer orders, involving activities such as receiving requests, checking inventory, processing payments, and shipping products. In computing more broadly, a process instance may refer to the execution of a workflow, program, or algorithm. The processes a system can perform collectively define its *behavior*.

Table 1: Example event data.

Case	Activity	Timestamp
1	a	1
2	a	2
3	a	3
3	c	4
4	b	5
2	b	6
5	b	7
5	a	8
1	b	9
2	d	10
1	c	11
3	b	12
2	c	13
4	a	14
4	c	15
5	d	16
5	c	17
4	d	18
1	d	19
3	d	20

Events that record which activity generated them, that is, events with a value for the attribute *activity* $\in \mathcal{A}$, can be associated with the process instances in which those activities occurred. In conventional process discovery [8], the attribute *case* $\in \mathcal{A}$ is used to correlate events with their corresponding process instances. Specifically, all events that share the same *case identifier* form the complete set of events for a single process instance, or *case*. Ordering these events by their timestamps using the total order $\leq$ yields a *trace*. A collection of traces is commonly referred to as an *event log*.

For instance, the events in Table 1 define five traces that can be grouped into event log $L = [\langle \mathsf{a},\mathsf{b},\mathsf{c},\mathsf{d}\rangle, \langle \mathsf{a},\mathsf{b},\mathsf{d},\mathsf{c}\rangle, \langle \mathsf{a},\mathsf{c},\mathsf{b},\mathsf{d}\rangle, \langle \mathsf{b},\mathsf{a},\mathsf{c},\mathsf{d}\rangle, \langle \mathsf{b},\mathsf{a},\mathsf{d},\mathsf{c}\rangle]$. For simplicity, we denote each trace as a sequence of activities without explicitly showing timestamps or case identifiers. In this example, these attributes can be inferred unambiguously from the activity sequences. In general, an event log may contain multiple traces that yield the same activity sequence. Hence, event logs are commonly conceptualized as multi-sets of activity sequences. More recently, object-centric process discovery [7] has been introduced, where events are grouped into process instances based on sets of attribute values that identify the objects involved in the activities that triggered the events.

2.4 Process Models

A *process model* describes the possible orderings of activities, providing a simplified representation of a process constructed for a specific purpose and target audience. It

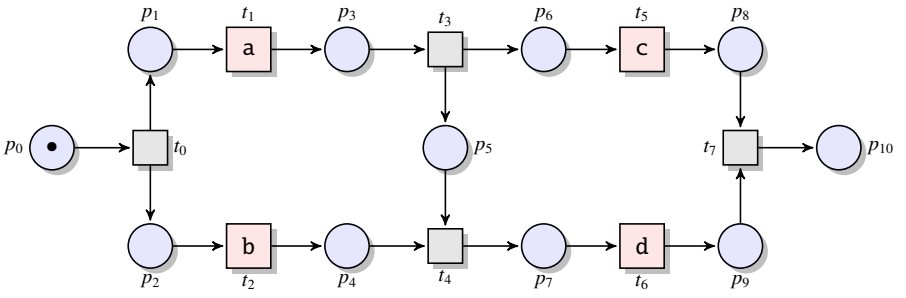

Fig. 1: A Petri net.

captures multiple process instances that aim to achieve a specific goal by following common activity patterns. For example, Petri nets [31] offer a formal modeling framework capable of representing concurrency, choice, and synchronization between activities, whereas directly-follows graphs (DFGs) [5] and stochastic directed action graphs (SDAGs) [10] provide abstract representations that emphasize directly-follows relations observed between recorded activities.

Figure 1 depicts an example Petri net consisting of eleven places (circles p_0 to p_{10}) and eight transitions (squares t_0 to t_7). Among these, four transitions (t_1, t_2, t_5, and t_6) are observable and represent the activities a, b, c, and d, respectively. These are the transitions whose execution is visible to an external observer. The remaining four transitions are silent, capturing internal activities that are not visible to observers. By projecting execution sequences of transitions described by the net onto the activities associated with the observable transitions, one obtains the activity sequences of the traces supported by the net. The Petri net shown in Fig. 1 supports exactly the five activity sequences induced by the traces in the event log L introduced in Section 2.3.

2.5 Definition

Let $\mathcal{E}$ be a universe of events and let $\mathcal{M}$ be a universe of process models. A technique that solves the process discovery problem can be conceptualized as a function d that maps collections of events and configurations C to process models, that is:

$$d : \mathcal{P}(\mathcal{E}) \times C \to \mathcal{M}. \tag{1}$$

A *configuration* of a discovery technique customizes the construction of the resulting process model. Most often, the configuration specifies the level of detail about the process that generated the input data that should be preserved in the discovered process model. In many cases, the configuration determines the level of detail from the process that generated the input data that should be retained in the discovered model. This level of detail can be adjusted by the user, for example, by using a slider control [30].

Discovered process models serve as a starting point for understanding and improving the processes of the system that produced the event data. They provide valuable insights for diverse stakeholders, including system analysts diagnosing inefficiencies, managers optimizing operations, and domain experts ensuring compliance and quality, as they uncover how the system behaves in the real world.

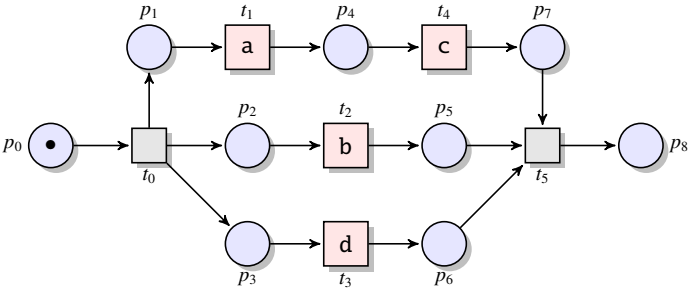

Fig. 2: A Petri net discovered from event log L introduced in Section 2.3 using the implementation of the Inductive Miner infrequent algorithm in ProM 6.11.

For example, Fig. 2 shows a Petri net discovered from the event log L introduced in Section 2.3. The model was obtained using the implementation of the Inductive Miner infrequent algorithm [22] in ProM 6.11 [37], configured with a noise threshold of 0.0.

3 Variants

A process model constructed as a solution to the original process discovery problem described in Section 2.5 aims to capture the full behavior of the system that produced the data, including behavior that has not been observed but is allowed by its rules. Such a model is useful for comprehensive system analysis. It enables analysts and engineers to reason about system behavior, identify design deficiencies, and verify that the system supports its intended functionalities. This section discusses several common variants of the conventional process discovery problem that differ in the representational focus of the models derived from the input event data.

3.1 Relevant Behavior Discovery

Often, only a portion of the system's behavior that is actively used and likely to be used in the future is of practical interest. Different organizations may use the same system in different ways, depending on their operational needs. Consequently, they may also have no practical interest in certain process instances that the system supports but that do not occur in their day-to-day operations. The focus of the *relevant behavior discovery* variant of the discovery problem is therefore on constructing models that represent this operationally active subset of behavior. As a result, the discovered models exclude obsolete or infrequent behavior and provide a clearer reflection of how the system functions in practice. Such models are useful for process optimization, resource planning, and continuous improvement because they emphasize dominant and recurring behavior patterns supported by empirical evidence.

3.2 Observed Behavior Discovery

The *observed behavior discovery* variant of the discovery problem emphasizes the importance of the recorded event data. When discovering the observed system's behav-

ior, the objective is to construct process models that accurately reproduce the system's behavior as captured in the event data, without inferring unobserved or hypothetical behavior. Such models are especially valuable for auditing, compliance checking, and retrospective analysis, where the focus is on verifying that actual executions conform to expectations or regulations. Observed behavior discovery prioritizes descriptive accuracy and minimizes assumptions beyond what is explicitly present in the data.

3.3 Process Forecasting

The *process forecasting* variant of the process discovery problem aims to construct models that describe the system's expected behavior in a given future period [27, 43], for example, in the next year. Rather than focusing on historical operations or on unconstrained generalizations of what the system can do in principle, this approach generalizes from historical data to predict how the system will behave under future conditions. Process forecasting enables proactive decision-making, supporting the preparation, adaptation, or redesign of processes before they take place. Such models are instrumental for capacity planning, risk management, and process evolution, where anticipating future trends and deviations is critical.

4 Challenges

Process discovery presents a range of challenges. Event data are often heterogeneous, incomplete, or noisy, while the underlying processes can be complex and subject to change over time. The main categories of challenges in process discovery are outlined below in an order that reflects their perceived importance.

4.1 Data Quality

Real world event data often contain missing attribute values, duplicate or spurious events, incorrect timestamps, or inconsistently named attributes. These data quality issues are commonly grouped under the broad umbrella of *noise* in event data. Typical sources of noise include human errors or deviations from prescribed behavior, asynchronous logging across distributed system components, and inconsistencies introduced by heterogeneous system integrations. The corresponding manifestations of noise include insertion, absence, ordering, and substitution noise [21]. Poor data quality undermines the reliability of discovered process models, potentially leading to misleading interpretations of the underlying system and, consequently, to suboptimal redesign and improvement decisions. Therefore, data cleaning, filtering, imputation, and semantic alignment constitute essential data preprocessing steps for meaningful discovery. Addressing noise in event data remains a non-trivial task, as effective mitigation strategies are often domain-specific and dependent on contextual process knowledge.

4.2 Data Incompleteness

An event log typically captures only a fraction of all process instances the system can perform. Specific instances or exceptional behaviors may be absent because they occur

rarely or were not recorded. In addition, event data does not naturally contain information about negative process instances, that is, instances that are known not to belong to the process, as they never occur and hence are not recorded. Consequently, discovered models risk undergeneralization, failing to represent legitimate but unseen process paths. In addition, the absence of negative examples is known to prevent the exact inference of many interesting classes of behavior [18]. Addressing incompleteness, therefore, requires designing discovery algorithms that generalize well from limited evidence while avoiding overfitting. Techniques such as probabilistic inference [10], simulation [3], and bootstrapping [29] can improve robustness by estimating or sampling plausible, yet unseen, behavior.

4.3 Representational Bias

Each discovery algorithm embodies assumptions about how processes should be represented, e.g., as imperative models, declarative rules, agent-based models, or causal graphs. These assumptions determine which behavioral patterns can be expressed in the discovered models and which are inherently excluded. On the one hand, restricting the set of admissible model patterns narrows the solution space of the discovery problem; on the other hand, it prevents the modeling of certain classes of processes [1]. Distinguishing concurrency from alternative choices, or rework loops from routine repetitions, is often difficult based solely on observed event data, since logs may not provide sufficient evidence to support a particular interpretation. Certain behavioral patterns, such as the one shown in Fig. 1, do not admit block-structured imperative representations that preserve the original concurrency relations, while other concurrency patterns require duplicate activities to obtain an equivalent block-structured form [28]. Moreover, free-choice models are unable to adequately capture long-term dependencies among activities [20, 38]. Selecting an appropriate process representation and paradigm, therefore, strongly influences both the interpretability and the utility of the discovered models and constitutes a non-trivial design decision in process discovery that needs to account for the concrete discovery technique and the characteristics of the discovered process.

4.4 Evaluation and Benchmarking

Assessing the performance of process discovery techniques remains a non-trivial task. Metrics for precision, fitness, generalization, and simplicity are well established for procedural models [34], but are far less mature for object-centric [9], data- and resource-aware [25], stochastic [11, 24], or agent-based [36] representations. Moreover, existing benchmark datasets are often limited to a few domain-specific logs, which limits the generalizability of empirical findings. Inconsistent log formats, preprocessing choices, and configuration settings across studies further hinder reproducibility. A key challenge for the community is, therefore, to establish standardized evaluation frameworks and open benchmarks that enable fair, transparent, and repeatable assessments across diverse modeling paradigms.

4.5 Multiple Perspectives

Modern event logs contain far more than control-flow information. They also record resource involvement, data attributes that influence process decisions, and case-level contextual information. Integrating these perspectives to discover models that jointly explain who performs activities, under which conditions, and with what outcomes remains a substantial challenge. Simple extensions of control-flow discovery often fail to capture complex data dependencies, conditional routing, or resource interactions. Multi-perspective discovery aims to unify (subsets of) behavioral, stochastic, organizational, informational, and transactional aspects within a single model, but achieving this requires richer semantics and more expressive modeling formalisms.

4.6 Scalability

Process discovery must increasingly handle event data that span millions of cases and billions of events. Efficient algorithms and scalable infrastructures are therefore essential, particularly when discovery results are expected to be produced close to real time. As data volume and complexity grow, both model discovery and quality assessment become more computationally demanding in terms of runtime and memory usage. Techniques such as data sampling, distributed processing, and incremental or streaming discovery can mitigate scalability challenges, although often at the cost of reduced accuracy or completeness. Balancing scalability with analytical depth thus remains an ongoing research problem. Finally, only a limited number of existing discovery techniques explicitly report their runtime requirements, which further complicates systematic comparison and evaluation.

4.7 Concept Drift

Real world processes evolve over time due to policy changes, system updates, or organizational adaptations. This phenomenon is known as concept drift. Concept drift causes event data to reflect a mix of outdated and current behaviors. Discovering a single static model from such data may obscure important temporal dynamics. Detecting and handling drift requires identifying change points, comparing models across time windows, and supporting incremental updates [32, 40]. Adaptive and online discovery techniques are increasingly crucial to ensure that models remain accurate and relevant.

4.8 Historical Dependencies

Event data often contain historical dependencies, where the execution of an activity depends on conditions or events that occurred far earlier in the process. Such dependencies may span long intervals, be mediated through data attributes, or arise from implicit business rules that are not directly observable in the data. Identifying these relationships from raw event data is challenging because the evidence needed to justify them may be sparse, weakly correlated, or confounded by unrelated, unregistered behavior. As a result, traditional discovery techniques that rely primarily on local or short-range patterns may overlook long-term interactions, leading to models that underrepresent the

true constraints of the process. Accurately capturing historical dependencies, therefore, remains an open research problem that requires richer representations, more expressive analysis techniques, and stronger statistical foundations [23].

4.9 Event Correlation

Identifying which events belong to the same process instance, or case, is a fundamental challenge in process mining. While traditional event logs provide explicit case identifiers, many modern information systems generate multi-object or object-centric data in which activities involve several interacting entities, e.g., orders, customers, and invoices. Inferring the correct correlations between events and process instances in such settings requires reasoning about object relationships, shared attributes, and temporal dependencies [14]. Wrong event correlation can lead to fragmented traces or the merging of unrelated behaviors, both of which severely distort the outcomes of process discovery and subsequent analyses.

4.10 Data Granularity

Data granularity may vary substantially within and across data sources, complicating its interpretation. Some systems record fine-grained technical events, while others capture only high-level business activities, and changes in system configuration or process implementation can further alter what is recorded. Such variability can lead to incomplete, inconsistent, or ambiguous behavioral evidence, making it challenging to reconstruct precise execution paths or infer accurate control-flow relationships. Discovery algorithms that assume consistent event abstraction may misinterpret coarse-grained data or fragment already fine-grained ones, resulting in distorted or misleading models. Managing heterogeneity in logging granularity, therefore, requires techniques that adapt to different levels of abstraction and reason robustly under partial or unevenly detailed observations [41], including techniques for multi-level, hierarchical discovery across abstraction layers.

4.11 Semantic and Contextual Interpretation

Event data typically record observed activities while omitting the contextual and semantic information that explains why those activities occur. Consequently, discovered models often describe processes syntactically but may lack interpretability for domain experts. Bridging this gap requires the incorporation of semantic knowledge, such as domain ontologies, goal hierarchies, and data dependencies, into the discovery process. Automatically linking low-level events to higher-level concepts or organizational semantics remains difficult when only conventional event logs are available. Effective integration of contextual knowledge would enable models that accurately represent the process and are semantically meaningful, improving their usefulness for explanation, compliance, and reasoning tasks.

4.12 Human-in-the-Loop and Explainability

While process discovery aims to automate model construction, human expertise remains essential for interpreting, validating, and refining the resulting models. Many existing algorithms operate as black boxes, producing complex models whose structure and underlying assumptions are difficult to understand or justify. Incorporating user feedback during discovery, for instance, through interactive visual analytics, constraint specification, or iterative model refinement, can improve both accuracy and user trust [33]. Designing explainable discovery techniques that make their reasoning transparent, for example, by exposing intermediate decisions or highlighting evidence supporting discovered patterns, is an emerging and increasingly important research direction.

5 Related Problems

This section examines related inference problems studied in adjacent fields, highlighting conceptual parallels and methodological insights relevant to process discovery.

5.1 Grammatical Inference

Grammatical inference and process discovery share striking conceptual similarities [10, 19]. Both aim to reconstruct an underlying generative system from observed examples. These examples are words or sentences in grammatical inference and process traces in process discovery. In each setting, the observed data provide positive evidence of system behavior, from which the learner seeks to infer a compact and general model capable of reproducing and predicting similar observations. Both fields must cope with noise and exceptional instances that deviate from the dominant patterns, and they develop techniques to mitigate such deviations without overfitting. They also emphasize representational parsimony. The inferred grammar or process model should capture the observed examples with high fidelity while remaining as simple as possible, in line with the Occam's razor principle. Similarly, both seek predictive power. Grammatical inference estimates the probability of the next character or word, while process discovery forecasts the likelihood of the next activity, trace suffix, or entire trace. Finally, both aspire to generalization by moving beyond the finite samples in the training data and characterizing the broader, unseen language or process that produced them. The main differences arise from the nature and structure of the input data, which are natural language text for grammatical inference and business process event traces for discovery.

5.2 Specification Discovery

Specification discovery in software engineering shares many conceptual foundations with process discovery [12, 26]. Both seek to reconstruct a formal description of system behavior from observed executions. In specification discovery, the input consists of program traces or logs of software runs, from which the goal is to infer behavioral specifications, such as temporal properties, state machines, or pre- and post-conditions, that characterize how the software behaves. Like process discovery, specification discovery

learns from positive examples, identifies and filters out noise or outlier traces, and aims to derive concise, human-interpretable models that generalize beyond the observed executions. Inferred specifications can also support predictive reasoning, for example, by signaling likely future violations or anticipating potential deviations. Both fields value simplicity and explanatory clarity, favoring minimal models that remain faithful to the data. Ultimately, both process and specification discovery aim to transform empirical observations into formal and generalizable descriptions that capture the essential behavior of the underlying system.

5.3 Model Synthesis

Model synthesis encompasses a family of techniques that aim to construct formal system models from behavioral observations. Within this paradigm, region theory in Petri nets [13, 15] and automata synthesis [16] in formal methods represent two prominent approaches. Both start from an observed behavior, for example, a transition system, execution log, or collection of traces, and infer a structural model that could have generated it. The goal is to reverse-engineer the causal or ordering relations underlying the observed behavior and express them as an interpretable and verifiable model. Region theory provides the mathematical foundation for synthesizing Petri nets from labeled transition systems. It identifies subsets of states, called regions, that correspond to potential places in the resulting net, ensuring that the constructed model reproduces the behavior of the original system. Automata synthesis follows a similar logic by deriving state machines or temporal automata from examples, formal specifications, or observed executions, often guided by constraints of consistency and minimality.

5.4 Other Inference Problems

Several other problems share conceptual foundations with process discovery. Automata learning studies how to construct behavioral models from observations and has produced influential techniques for learning regular and symbolic automata. Inductive logic programming and constraint mining aim to infer declarative rules from examples, a setting closely related to the discovery of declarative process models. In data mining, sequential pattern mining and episode mining identify frequent behavioral patterns that serve as the basis for some process discovery methods. System identification in control theory similarly seeks to infer state-based dynamic models from time-series observations and addresses issues of noise, generalization, and prediction. Finally, causal discovery investigates how to recover causal dependencies from observational data, an increasingly important concern in predictive and prescriptive process analytics.

6 Quality

The success of process discovery depends on the quality of the resulting models. Thus, discovered models should be assessed against well-defined and practically meaningful quality criteria that determine their adequacy for analysis and decision-making.

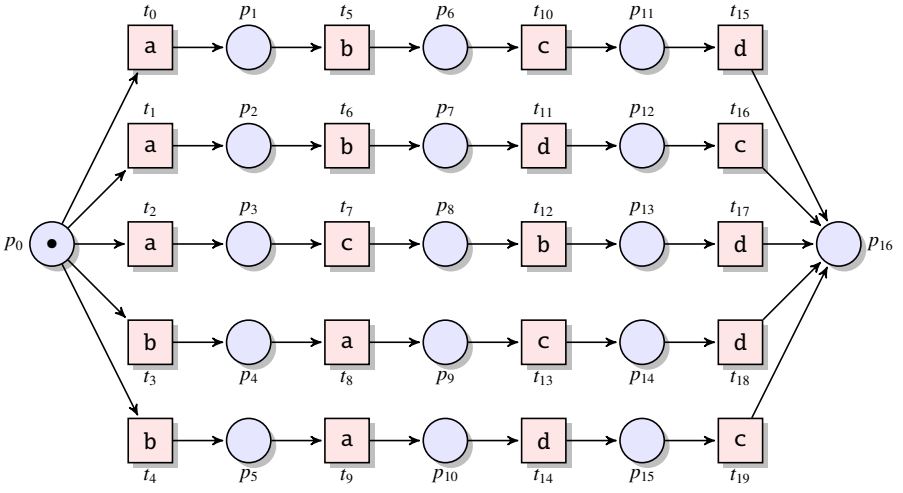

Fig. 3: All traces Petri net.

In the conventional process discovery, where discovered process models aim to describe event traces the system can generate, four interrelated quality dimensions are widely accepted [17]. A discovered model should allow as few traces as possible that are not present in the log (good *precision*), should describe as many as possible of the traces recorded in the log (good *recall*, also known as *fitness*), should allow traces that may stem from the same process but are not present in the log (good *generalization*), and should be as simple as is consistent with the other goals (good *simplicity*). Many desirable properties for the measurements of these quality dimensions [4] and techniques to implement the measurements [34] have been proposed.

These dimensions are not independent. For instance, increasing precision often comes at the expense of generalization, while maximizing fitness can lead to unnecessarily complex models. Process discovery is fundamentally a multi-objective optimization problem [17, 42] that shapes both algorithm design and the interpretation of discovered models. The relative weighting of these quality dimensions depends on the purpose and audience of the model. Managers may prioritize simplicity and generalization for decision support (that is, relevant behavior discovery and process forecasting, cf. Sections 3.1 and 3.3), whereas system engineers may prefer fitness and precision for system diagnostics (that is, observed behavior discovery, cf. Section 3.2). Each quality dimension is usually measured on a scale from zero to one, where larger values indicate better model quality.

Figures 3 and 4 illustrate two additional models that can be discovered from the event log L. All models shown in Figs. 1 to 4 exhibit perfect recall with respect to L, since each of them describes all the traces recorded in this event log. The models depicted in Figs. 1, 3 and 4 also exhibit perfect precision, since they do not allow for any traces that are absent from L. In contrast, the model in Fig. 2 does not have perfect precision, as it allows for traces that do not occur in the event log, namely $\langle \mathsf{a},\mathsf{c},\mathsf{d},\mathsf{b}\rangle$, $\langle \mathsf{a},\mathsf{d},\mathsf{b},\mathsf{c}\rangle$, $\langle \mathsf{a},\mathsf{d},\mathsf{c},\mathsf{b}\rangle$, $\langle \mathsf{b},\mathsf{d},\mathsf{a},\mathsf{c}\rangle$, $\langle \mathsf{d},\mathsf{a},\mathsf{b},\mathsf{c}\rangle$, $\langle \mathsf{d},\mathsf{a},\mathsf{c},\mathsf{b}\rangle$, and $\langle \mathsf{d},\mathsf{b},\mathsf{a},\mathsf{c}\rangle$. The models depicted

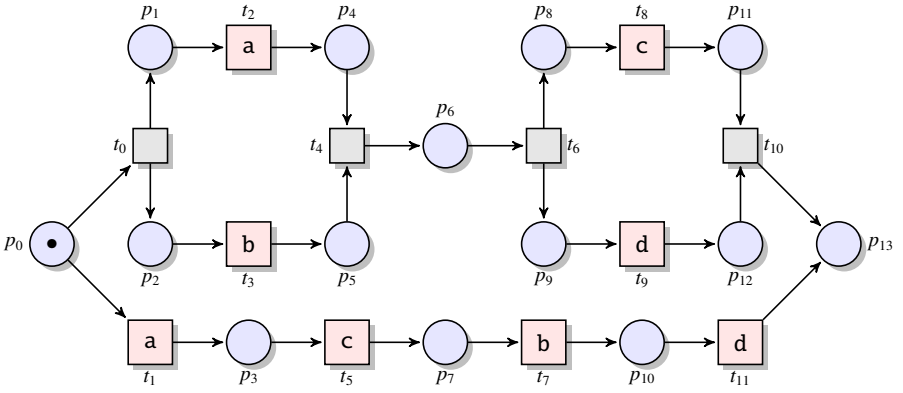

Fig. 4: A block structured Petri net with duplicate activities.

in Figs. 1, 2 and 4 are simpler than the one in Fig. 3 if the number of nodes and arcs is used as a measure of model complexity. Finally, the additional traces permitted by the model in Fig. 2 can be interpreted as instances of generalization.

Alignments are diagnostic artifacts that describe how a process model explains the traces recorded in an event log [6]. An alignment provides a step-by-step correspondence between a log trace and a model execution, identifying where the model can reproduce the observed activities and where deviations occur. Optimal alignments minimize the cost of deviation steps. Intuitively, the lower the cost of alignments between the log traces and the model, the better the precision and recall of the model.

The SOLID-M framework provides an ontology-aware approach to assessing the quality of conceptual models discovered from event data [36]. It extends traditional quality measures grounded in log and model traces to support new process discovery types studied within Object-Centric Process Mining and Agent System Mining. SOLID-M supports the assessment of model quality by how well the model describes the structure and behavior of the system that generated the data, where correspondences between the model, system, and data elements are traced through a shared ontology.

7 Conclusion

Process discovery has made significant progress over the past two decades. Advances in algorithms, tools, and theoretical foundations have expanded the range of systems that can be analyzed based on their event data. Despite this progress, many open gaps remain. Comprehensive benchmarks that integrate both real world and synthetic datasets are lacking. As a result, process discovery results are rarely comparable, since they are evaluated on different datasets using diverse quality measures. Current discovery techniques tend to exhibit strong representational biases, most notably toward block-structured models without duplicate activities, which limits their ability to reconstruct more complex process behaviors, such as those illustrated in Figs. 1 and 4. Moreover, the community lacks large-scale, publicly available event logs for systematic evaluation, and there is little agreement on which model quality measures should be adopted for

broad use [34]. Among the standard quality dimensions, generalization remains perhaps the least understood and most neglected aspect of discovery quality. Empirical evaluations involving ground-truth systems, where algorithms are assessed based on their ability to rediscover known processes, are exceedingly rare [39]. Equally underexplored are the qualities of the discovery algorithms themselves, including their robustness and scalability, rather than merely the quality of their outputs. As the field continues to mature, addressing these open issues will be essential for establishing process discovery as a rigorous, reproducible, and insightful scientific discipline. The years ahead will bring research that narrows these gaps and deepens our understanding of how systems can be accurately discovered from the traces of their behavior.

References

[1] van der Aalst, W.M.P.: On the representational bias in process mining. In: WETICE, pp. 2–7, IEEE Computer Society (2011), https://doi.org/10.1109/WETICE.2011.64
[2] van der Aalst, W.M.P.: Process Mining—Data Science in Action. Springer, 2nd edn. (2016), https://doi.org/10.1007/978-3-662-49851-4
[3] van der Aalst, W.M.P.: Process mining and simulation: A match made in heaven! In: SummerSim, pp. 4:1–4:12, ACM (2018)
[4] van der Aalst, W.M.P.: Relating process models and event logs—21 conformance propositions. In: ATAED, CEUR Workshop Proceedings, vol. 2115, pp. 56–74, CEUR-WS.org (2018), URL http://ceur-ws.org/Vol-2115/ATAED2018-56-74.pdf
[5] van der Aalst, W.M.P.: A practitioner's guide to process mining: Limitations of the directly-follows graph. Procedia Comput. Sci. **164**, 321–328 (2019), https://doi.org/10.1016/j.procs.2019.12.189
[6] van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. WIREs Data. Mining. Knowl. Discov. **2**(2), 182–192 (2012), https://doi.org/10.1002/WIDM.1045
[7] van der Aalst, W.M.P., Berti, A.: Discovering object-centric Petri nets. Fundam. Informaticae **175**(1–4), 1–40 (2020), https://doi.org/10.3233/FI-2020-1946
[8] van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004), https://doi.org/10.1109/TKDE.2004.47
[9] Adams, J.N., van der Aalst, W.M.P.: Precision and fitness in object-centric process mining. In: ICPM, pp. 128–135, IEEE (2021), https://doi.org/10.1109/ICPM53251.2021.9576886
[10] Alkhammash, H., Polyvyanyy, A., Moffat, A.: Stochastic directly-follows process discovery using grammatical inference. In: CAiSE, LNCS, vol. 14663, pp. 87–103, Springer (2024), https://doi.org/10.1007/978-3-031-61057-8_6
[11] Alkhammash, H., Polyvyanyy, A., Moffat, A., García-Bañuelos, L.: Entropic relevance: A mechanism for measuring stochastic process models discovered from event data. Inf. Syst. **107**, 101922 (2022), https://doi.org/10.1016/J.IS.2021.101922
[12] Ammons, G., Bodík, R., Larus, J.R.: Mining specifications. In: POPL, pp. 4–16, POPL02, ACM (2002), https://doi.org/10.1145/503272.503275
[13] Badouel, É., Bernardinello, L., Darondeau, P.: Petri Net Synthesis. Springer (2015), https://doi.org/10.1007/978-3-662-47967-4
[14] Bayomie, D., Ciccio, C.D., Mendling, J.: Event-case correlation for process mining using probabilistic optimization. Inf. Syst. **114**, 102167 (2023), https://doi.org/10.1016/J.IS.2023.102167

[15] Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Process mining based on regions of languages. In: BPM, LNCS, vol. 4714, pp. 375–383, Springer (2007), https://doi.org/10.1007/978-3-540-75183-0_27
[16] Biermann, A.W., Feldman, J.A.: On the synthesis of finite-state machines from samples of their behavior. IEEE Trans. Computers **21**(6), 592–597 (1972), https://doi.org/10.1109/TC.1972.5009015
[17] Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. Int. J. Cooperative Inf. Syst. **23**(01), 1440001 (2014), https://doi.org/10.1142/S0218843014400012
[18] Gold, E.M.: Language identification in the limit. Information and Control **10**(5), 447–474 (1967), https://doi.org/10.1016/s0019-9958(67)91165-5
[19] de la Higuera, C.: Grammatical Inference: Learning Automata and Grammars. Cambridge University Press (2010)
[20] Kalenkova, A., Carmona, J., Polyvyanyy, A., Rosa, M.L.: Automated repair of process models with non-local constraints using state-based region theory. Fundam. Informaticae **183**(3–4), 293–317 (2021), https://doi.org/10.3233/FI-2021-2089
[21] Karunaratne, A., Polyvyanyy, A., Moffat, A.: The effects of log noise in process mining. IEEE Access **13**, 198540–198563 (2025), https://doi.org/10.1109/access.2025.3635682
[22] Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: BPM Workshops, LNBIP, vol. 171, pp. 66–78, Springer (2013), https://doi.org/10.1007/978-3-319-06257-0_6
[23] Leemans, S.J.J., Mannel, L.L., Sidorova, N.: Significant stochastic dependencies in process models. Inf. Syst. **118**, 102223 (2023), https://doi.org/10.1016/J.IS.2023.102223
[24] Leemans, S.J.J., Polyvyanyy, A.: Stochastic-aware precision and recall measures for conformance checking in process mining. Inf. Syst. **115**, 102197 (2023), https://doi.org/10.1016/J.IS.2023.102197
[25] de Leoni, M., van der Aalst, W.M.P., van Dongen, B.F.: Data- and resource-aware conformance checking of business processes. In: BIS, LNBIP, vol. 117, pp. 48–59, Springer (2012), https://doi.org/10.1007/978-3-642-30359-3_5
[26] Lo, D., Khoo, S.C., Han, J., Liu, C.: Mining Software Specifications: Methodologies and Applications. CRC Press, Inc. (2017)
[27] Poll, R., Polyvyanyy, A., Rosemann, M., Röglinger, M., Rupprecht, L.: Process forecasting: Towards proactive business process management. In: BPM, LNCS, vol. 11080, pp. 496–512, Springer (2018), https://doi.org/10.1007/978-3-319-98648-7_29
[28] Polyvyanyy, A.: Structuring Process Models. Ph.D. thesis, University of Potsdam (2012), URL `http://opus.kobv.de/ubp/volltexte/2012/5902/`
[29] Polyvyanyy, A., Moffat, A., García-Bañuelos, L.: Bootstrapping generalization of process models discovered from event data. In: CAiSE, pp. 36–54, Springer (2022), https://doi.org/10.1007/978-3-031-07472-1_3
[30] Polyvyanyy, A., Smirnov, S., Weske, M.: Process model abstraction: A slider approach. In: EDOC, pp. 325–331, IEEE Computer Society (2008), https://doi.org/10.1109/EDOC.2008.17
[31] Reisig, W.: Understanding Petri Nets—Modeling Techniques, Analysis Methods, Case Studies. Springer (2013), https://doi.org/10.1007/978-3-642-33278-4
[32] Sato, D.M.V., Freitas, S.C.D., Barddal, J.P., Scalabrin, E.E.: A survey on concept drift in process mining. ACM Comput. Surv. **54**(9), 189:1–189:38 (2022), https://doi.org/10.1145/3472752
[33] Schuster, D.: Incremental Process Discovery, LNBIP, vol. 540. Springer (2025)
[34] Syring, A.F., Tax, N., van der Aalst, W.M.P.: Evaluating conformance measures in process mining using conformance propositions. Trans. Petri Nets Other Model. Concurr. **14**, 192–

221 (2019), https://doi.org/10.1007/978-3-662-60651-3_8
[35] Tour, A., Polyvyanyy, A., Kalenkova, A.: Agent system mining: Vision, benefits, and challenges. IEEE Access **9**, 99480–99494 (2021), https://doi.org/10.1109/ACCESS.2021.3095464
[36] Tour, A., Polyvyanyy, A., Kalenkova, A.: SOLID-M: An ontology-aware quality framework for conceptual models discovered from event data. Inf. Syst. **137**, 102641 (2026), https://doi.org/10.1016/J.IS.2025.102641
[37] Verbeek, E., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: ProM 6: The process mining toolkit. In: BPM Demonstration Track, CEUR Workshop Proceedings, vol. 615, CEUR-WS.org (2010)
[38] Wen, L., van der Aalst, W.M.P., Wang, J., Sun, J.: Mining process models with non-free-choice constructs. Data Min. Knowl. Discov. **15**(2), 145–180 (2007), https://doi.org/10.1007/S10618-007-0065-Y
[39] van der Werf, J.M.E.M., Polyvyanyy, A., van Wensveen, B.R., Brinkhuis, M., Reijers, H.A.: All that glitters is not gold: Four maturity stages of process discovery algorithms. Inf. Syst. **114**, 102155 (2023), https://doi.org/10.1016/J.IS.2022.102155
[40] Yeshchenko, A., Ciccio, C.D., Mendling, J., Polyvyanyy, A.: Visual drift detection for event sequence data of business processes. IEEE Trans. Vis. Comput. Graph. **28**(8), 3050–3068 (2022)
[41] van Zelst, S.J., Mannhardt, F., de Leoni, M., Koschmider, A.: Event abstraction in process mining: Literature review and taxonomy. Granular Computing **6**(3), 719–736 (2020), https://doi.org/10.1007/s41066-020-00226-2
[42] Zhian, H., Buyya, R., Polyvyanyy, A.: Multi-objective metaheuristics for effective and efficient stochastic process discovery. In: BPM, LNCS, vol. 16044, pp. 469–486, Springer (2025), https://doi.org/10.1007/978-3-032-02867-9_28
[43] Zhou, W., Polyvyanyy, A., Bailey, J.: Event data and process model forecasting. In: CAiSE Forum, LNBIP, vol. 520, pp. 3–10, Springer (2024), ISBN 9783031610004, ISSN 1865-1356, https://doi.org/10.1007/978-3-031-61000-4_1