SAPIENZA
UNIVERSITÀ DI ROMA

LTL_f Declarative Specifications of Process Behavior: Foundations, Expressiveness, and Applications

Department of Computer Science
Ph.D. in Computer Science (XXXVII cycle)

Luca Barbaro
ID number 1773371

Advisor
Prof. Claudio Di Ciccio

Academic Year 2025/2026

Thesis defended on May 11th, 2026
in front of a Board of Examiners composed by:

Prof. Barbara Re

Prof. Xixi Lu

Prof. Han van der Aa

LTL$_f$ Declarative Specifications of Process Behavior: Foundations, Expressiveness, and Applications

Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: luca.barbaro@uniroma1.it

Dedicated to my family

Extended Abstract

Motivation and rationale

Across a wide range of modern process-aware systems, behavioral knowledge is captured and used in multiple shapes. The same system may be described at design time through process modeling, observed at runtime through execution data, and acted upon through decision-making components that adapt to changing operational conditions. As a result, process representations must remain consistent across heterogeneous tasks, supporting verification, diagnostics, and optimization, without the need to redefine them from scratch every time the viewpoint changes.

Meeting these requirements calls for representations that are expressive enough to capture complex behavioral relations, while remaining susceptible to analytical reasoning, quantitative assessment, and integration with data-driven techniques. These challenges arise in a variety of domains, including business processes, service-oriented architectures, and autonomous and learning-based systems.

Among these landscapes, Business Process Management (BPM) provides a particularly illustrative domain in which the value of behavioral process knowledge becomes both operational and measurable. Business information systems generate large amounts of execution data that records how processes unfold in practice. This has fostered the development of *process mining* techniques, which aim to discover process representations from observed execution traces in the form of *event logs* and to assess whether observed behavior matches expected behavior. Consequently, process representations should not be regarded as mere design-time artifacts. Instead, they enable organizations to obtain a complete view of their operations, identify inefficiencies and potential deviations, and systematically improve the way work is performed. Indeed, behavioral insights directly translate into measurable performance gains, affecting key performance indicators (KPIs), namely throughput time, resource utilization, regulation compliance, and operational costs. Business process modeling (hereafter *process modeling*) has traditionally relied on *imperative* (or *procedural*) representations (e.g., Workflow net or BPMN) that explicitly define the allowed execution paths of a system. While these models

provide precise operational semantics and support automated execution, their explicit nature may restrict analysis and diagnosis in settings characterized by variability, flexibility, or partial observability. Conversely, *declarative specifications* have received increasing interest, as they suggest a significantly different way of representing processes, describing behavior in terms of constraints that executions must satisfy, offering a more flexible representation that is particularly suited to highly dynamic environments. These constraints are typically expressed as temporal rules grounded in logics such as Linear Temporal Logic on Finite Traces (LTL_f), stating *what* must hold in a process rather than prescribing *how* it is achieved. As a result, they can serve as an internal behavioral layer that can be reused across multiple analysis and reasoning tasks. Despite their complementary strengths, imperative and declarative paradigms are often treated as separate approaches, adjusting their adoption across different analytical and reasoning tasks.

Similar process representation challenges arise in service-based and cyber-physical settings, as well as in autonomous and learning-based systems, where agents must continuously adapt their behavior while still respecting safety and compliance constraints. In these scenarios, behavioral knowledge configures as a basis for monitoring and assessing whether observed or learned policies remain consistent with expected temporal and causal relations. This thesis investigates the use of declarative temporal specifications as an abstraction layer that span the full lifecycle of behavioral reasoning in process-aware systems. The main research objective guiding the work is stated as follows:

> **RO: Declarative Specifications for Process-Aware Systems**
>
> Investigate the use of LTL_f declarative specifications as an abstraction layer for representing, verifying, and exploiting behavioral knowledge in process-aware systems.

With this goal in mind, we set three research questions across process representations, verification, and decision making.

> **RQ1: Process Representation Paradigms**
>
> How can a declarative process specification be systematically derived from an imperative process model, preserving behavioral equivalence between the two representations?

First, we discuss the relationship between imperative and declarative process representations in the business processes domain, showing how a declarative specification can be systematically derived from an imperative model while preserving

behavioral equivalence. To this end, we introduce a systematic method to synthesize declarative specifications from safe and sound Workflow nets, together with formal guarantees ensuring that the two representations capture exactly the same behavior. This result enables principled transitions between paradigms, allowing analysts to select the most suitable representation depending on the task at hand.

> **RQ2: Conformance Measurement**
>
> How can the satisfaction of declarative process specifications be quantitatively assessed over observed executions?

Once a process has been modeled, the growing availability of execution data makes it natural to quantify how closely observed traces adhere to the normative representation. This is the goal of *conformance checking*, which measures the degree of correspondence between a model (or specification) and recorded executions. Traditional conformance checking verification techniques for declarative specifications often assess individual rules in isolation, overlooking their mutual interplay and offering primarily binary satisfaction outcomes. We address this gap by proposing a probabilistic framework to quantify the satisfaction of declarative process specifications over event logs. We introduce interestingness measures that characterize the degree to which LTL_f declarative constraints are supported by observed behavior. These measures enable behavioral diagnostics and provide a foundation for detecting deviations, changes, and emerging patterns in process executions.

> **RQ3: Decentralized Data Sources**
>
> How can declarative specifications be checked against execution data that is distributed across multiple independent sources?

Capturing specification-level insights becomes even more relevant when execution data are distributed across independent sources, each party observing only a partial view of the process, while a global assessment must be derived from distributed evidence. In these inter-organizational settings, traditional conformance checking approaches are hindered by confidentiality and governance constraints that prevent centralizing raw event logs. To overcome this limitation, we propose CONFINE, a secrecy-preserving framework that enables process mining and declarative conformance checking without disclosing sensitive execution data between collaborating organizations. CONFINE executes mining and checking algorithms inside trusted execution environments (TEEs), where event logs are transmitted and processed under attested confidentiality guarantees. The architecture supports a symmetric collaboration model in which each party can provide execution data and perform joint analyses, enabling behavioral insight to cross organizational boundaries while

preserving data sovereignty.

Project support and interpretation lens

This work has been framed around the project *"Predictive process monitoring for production planning"* (PPMPP), funded by the Latium Region under the PO FSE+ programme (grant B83C22004050009) and co-funded by Slim Aluminium SpA.[1]

The project was carried out in a large-scale manufacturing setting in which operational workflows exhibit a structured and repetitive framework, as typical of production environments (e.g., machine-intensive lines, batch processing phases, and quality control procedures). Yet, their actual execution is heavily defined by context-dependent constraints that go far beyond a fixed prescriptive control-flow. In particular, production planning decisions, maintenance needs (both scheduled and unforeseen), and the coordination of multiple organizational actors and responsibilities continuously affect which actions are feasible, when they can be performed, and under which conditions. As a result, even when the underlying routine remains stable, executions are subject to constraints that reflect operational policies, normative rules, and interactions across parties, among others.

This context provided the ground for RO of this thesis. Instead of sticking to a single modeling paradigm, the need emerges for a complementary behavioral representation that can *(i)* coexist with highly structured process models, *(ii)* explicitly captures contextual requirements as constraints, and *(iii)* support verification and exploitation of behavioral knowledge over observed executions. Yielding these capabilities motivates our research objective, i.e., investigating declarative specifications for representing, verifying, and exploiting behavioral knowledge in process-aware systems. The idea is to shift from a purely imperative control-flow view to a constraint-based perspective of process behavior, in which operational requirements can be stated explicitly over time. Such an abstraction remains compatible with structured process models, supports verification over observed executions, and can be leveraged to guide adaptive decision making in the presence of variability and unforeseen scenarios.

In detail, RQ1 addresses the need for a behavioral representation that supports transitions between structured imperative process representations and constraint-based declarative specifications; RQ2 targets quantitative assessments of whether operational and regulatory constraints are actually respected during in-house executions; RQ3 reflects the distributed, multi-party nature of production ecosystems and their confidentiality requirements.

[1]Funding: "Ricerca e innovazione nel Lazio – incentivi per i dottorati di innovazione per le imprese e per la PA – L.R. 13/2008".

Research contribution

For ease of reference, we list below the publications and accompanying software implementations and artifacts that constitute the main scientific outputs of this thesis. Each entry is followed by a pointer to the corresponding research question (RQ1 to RQ3) addressed in this dissertation.

[26] Barbaro, L.; Varricchione, G.; Montali, M.; Di Ciccio, C. *From Sound Workflow Nets to LTL$_f$ Declarative Specifications by Casting Three Spells.* BPM Forum 2025, LNBIP 564, pp. 3–22, Springer, 2025. doi: 10.1007/978-3-032-02929-4_1
Research contribution RQ1

Software implementation: `github.com/l2brb/Sp3llsWizard`

Author contribution: First author and primary contributor. Led the conceptualization and formalization of the approach, algorithm design, implemented the software artifact, and conducted the experimental evaluation.

[46] Cecconi, A.; Barbaro, L.; Di Ciccio, C.; Senderovich, A. *Measuring rule-based LTL$_f$ process specifications: A probabilistic data-driven approach.* Information Systems, 120:102312, 2024. doi: 10.1016/j.is.2023.102312
Research contribution RQ2

Replication package:
`github.com/l2brb/Measurement-change-point-evaluation`

Author contribution: Contributed to the experimental evaluation, including implementation support, testing activities, and empirical assessment.

[94] Goretti, V.; Basile, D.; Barbaro, L.; Di Ciccio, C. *Trusted Execution Environment for Decentralized Process Mining.* CAiSE 2024, LNCS 14663, pp. 509–527, Springer, 2024. doi: 10.1007/978-3-031-61057-8_30
Research contribution RQ3

[93] Goretti, V.; Basile, D.; Barbaro, L.; Di Ciccio, C. *CONFINE: Preserving Data Secrecy in Decentralized Process Mining.* ICPM 2024 Doctoral Consortium & Demo Track, CEUR WS Proc. 3783, 2024. https://ceur-ws.org/Vol-3783/paper_324.pdf
Research contribution RQ3

[30] Basile, D.; Goretti, V.; Barbaro, L.; Reijers, H. A.; Di Ciccio, C. *A TEE-based approach for preserving data secrecy in process mining with decentralized sources* Journal of Information Security and Applications, Volume 98, 2026. doi: 10.1016/j.jisa.2026.104381
Research contribution RQ3

Software implementation: github.com/Process-in-Chains/CONFINE

Author contribution: Contributed to the literature analysis and application scenario definition, and participated in the implementation, testing, and experimental evaluation of the framework.

Additional contributions

Besides the research activity relating to the work of this thesis, we also contributed to parallel work carried out within the project *"Distributed Ledger and Credit Guarantee Schemes"*, in collaboration with the Departments of Computer Science and Management of Sapienza University of Rome. For completeness, we report the resulting publications here, although the topics they address are beyond the scope of this thesis. The first work investigates how Distributed Ledger Technologies (DLTs), and blockchain in particular, can address structural weaknesses of Credit Guarantee Schemes (CGSs), such as limited transparency, operational inefficiencies, and frictions in multi-party information exchange. We propose a conceptual framework that maps the credit guarantee lifecycle from application and onboarding to monitoring and potential enforcement to points where blockchain features (immutability, traceability, and smart-contract automation) can provide measurable benefits, including reduced fraud risk and streamlined coordination among borrowers, banks, and credit guarantee institutions. A key contribution is the discussion of blockchain configuration choices (access level, governance, and consensus) in light of confidentiality and institutional constraints, outlining design trade-offs and directions tailored to CGS processes.

[122] Leo, S.; Delle Foglie, A.; Barbaro, L.; Marangone, E.; Panetta, I.C.; Di Ciccio, C. *Transforming Credit Guarantee Schemes with Distributed Ledger Technology.* ICBT 2024. Lecture Notes in Networks and Systems, vol 1083. Springer. doi: 10.1007/978-3-031-67431-0_30

We extended the above line of research by introducing BLINK, a modular, policy-aligned blockchain framework specifically designed for credit guarantee institutions (CGIs). BLINK translates institutional requirements into concrete system-level requirements and architectural choices, and operationalizes them through workflow automation and verifiable document handling. We develop and compare two deployment variants and provide a prototype-based preliminary assessment, including gas-cost and service-time analyses across multiple EVM-compatible platforms, thereby complementing conceptual arguments with feasibility evidence. Overall, BLINK contributes a structured blueprint for modernizing CGI operations through auditable, interoperable, and confidentiality-aware digital infrastructures.

[123] Leo, S.; Delle Foglie, A.; Barbaro, L.; Marangone, E.; Panetta, I. C.; Di Ciccio, C. *BLINK: Blockchain-Linked Network for Credit Guarantee Institutions.* Financial Innovation 12, 82 (2026). doi: 10.1186/s40854-025-00880-y

Thesis Outline

Chapter 1 introduces the research context of this thesis, outlines the research objective addressed, and clarifies the scope and structure of the work.

Chapter 2 provides the background necessary to understand the contributions of the thesis. It introduces process modeling fundamentals, introducing the different representation paradigms, including Petri nets and Workflow nets, together with their execution semantics. The chapter then presents Linear Temporal Logic on finite traces (LTL_f) as a formal language for declarative process specifications, and concludes with the foundations of conformance checking and behavioral verification over event logs.

Chapter 3 addresses the problem of bridging imperative and declarative process representation paradigms, presenting the systematic method to synthesize LTL_f declarative specifications from safe and sound Workflow Nets, preserving the bisimilarity between the original models. The chapter provides formal basics, discusses implications, and reports on an empirical evaluation of the proposed translation.

Chapter 4 focuses on the quantitative analysis of declarative process specifications. It proposes a set of measures to assess the interestingness of LTL_f constraints with respect to the executions observed in the event logs. The chapter discusses the properties of the proposed measures and evaluates them on synthetic and real-world event logs.

Chapter 5 extends declarative conformance checking to settings where execution data are distributed across multiple sources, proposing a decentralized framework for securely aggregating partial observations and executing process mining algorithms without requiring centralized access to raw event logs. The chapter discusses architectural choices, security assumptions, performance concerns, and implications.

Chapter 6 concludes the thesis by summarizing the main results and outlining directions for future research that build upon the foundations established.

Contents

Extended Abstract i

1 Introduction 1

2 Background 7
- 2.1 Process Representations and Semantics . . . 7
  - 2.1.1 Petri Nets and Workflow Nets . . . 7
  - 2.1.2 Linear Temporal Logic on finite traces LTL_f . . . 11
  - 2.1.3 Declarative Specifications . . . 13
- 2.2 Event Logs and Behavior Verification . . . 16
  - 2.2.1 Event Logs . . . 16
  - 2.2.2 Process Mining . . . 16
  - 2.2.3 Conformance Checking . . . 17

3 From Imperative Process Models to LTL_f Declarative Specifications 19
- 3.1 Introduction . . . 19
- 3.2 Related Work . . . 20
- 3.3 Synthesize LTL_f Declarative Specifications from Workflow Nets . . . 21
- 3.4 Implementation and evaluation . . . 25
  - 3.4.1 Automata bisimulation . . . 26
  - 3.4.2 Performance analysis . . . 26
  - 3.4.3 Using constraints as determinants for process diagnosis . . . 29

4 Measuring LTL_f Process Specifications 31
- 4.1 Introduction . . . 31
- 4.2 Related Work . . . 33
- 4.3 Event logs and Linear Temporal Logic on Finite Traces LTL_f . . . 34
- 4.4 Reactive Constraints (RCons) and process specifications . . . 36
  - 4.4.1 Reactive Constraints (RCons) . . . 36

4.4.2 Process specifications . . . 37
4.5 Statistical preliminaries . . . 38
4.5.1 Basic Probability Notation . . . 38
4.5.2 Conditional Probability . . . 39
4.5.3 Discrete Random Variables . . . 39
4.5.4 Maximum Likelihood Estimation . . . 40
4.6 Interestingness measures for Reactive Constraint (RCon) . . . 40
4.7 Estimators for LTL$_f$ formulae . . . 42
4.7.1 Trace estimators . . . 42
4.7.2 Log estimators . . . 44
4.8 Evaluation and measurement of specifications . . . 45
4.8.1 Evaluating specifications . . . 45
4.8.2 Estimators for Specifications . . . 46
4.8.3 Computing Measures of Interestingness for Specifications . . . 49
4.8.4 Discussion . . . 50
4.9 Evaluation . . . 52
4.9.1 Measuring single rules and entire specifications . . . 53
4.9.2 Differing measures . . . 54
4.9.3 Impact of process drift on specification measures . . . 56
4.9.4 Discussion . . . 66

5 Decentralized Conformance Checking from Different Data Sources 67
5.1 Introduction . . . 67
5.2 Related Work . . . 69
5.3 Confidential Computing and Trusted Execution Environments . . . 71
5.4 Collaborative Scenario . . . 73
5.5 Design . . . 76
5.6 Realization . . . 78
5.6.1 Deployment . . . 78
5.6.2 The CONFINE protocol . . . 79
5.7 Evaluation . . . 83
5.7.1 Implementation . . . 83
5.7.2 Convergence . . . 85
5.7.3 Performance Assessment . . . 85
5.8 Discussion . . . 90
5.8.1 Practical Implications . . . 90

6 Conclusion and Future Research Directions **93**
6.1 Main findings 93
6.2 Future work 96

Bibliography **99**

List of Figures

2.1 A Workflow net 7
2.2 Reachability graph 10
2.3 Example FSAs 13

3.1 A graphical sketch of the algorithm 22
3.2 FSA equivalent to the specification 25
3.3 Transformation rules 27
3.4 Test results for incremental number of constraints 27
3.5 Incremental constraints dimension tests 28
3.6 Scatter plot for constraints clustered by fitness values 29

4.1 Discovered specification confidence 55
4.2 Help-Desk sub-log confidence values 58
4.3 Sepsis sub-log confidence values 59
4.4 Specification measure oscillations on the Sepsis event log 62
4.5 Specification measure oscillations on the help desk event log 63
4.6 Confidence of the erratic clusters with the Help-Desk log 65
4.7 Confidence of the erratic clusters with the Sepsis log 65

5.1 Collaborative Workflow net 73
5.2 The CONFINE high-level architecture 76
5.3 Secure Miner sub-components 77
5.4 UML deployment diagram of the CONFINE architecture 78
5.5 Unfolding example for the initialization, remote attestation and data transmission phases of the CONFINE protocol 80
5.6 Computation phase of the CONFINE protocol 82
5.7 HeuristicsMiner output 84
5.8 Memory usage test results 86
5.9 Segment size test result 87
5.10 Scalability test results 88

List of Tables

2.1 Semantics of some DECLARE constraint templates 13

3.1 Specification generated from the Workflow net 22
3.2 Performance comparison with real-world process models 29

4.1 Checking of a specification against a log. 35
4.2 Selection of interestingness measures 41
4.3 Selection of interestingness measures for specifications of RCons . . . 50
4.4 Example of a specification violated by high probability constraints. . 52
4.5 Details of the real-world event logs used for the evaluation 53
4.6 Measurements of sub-specifications 56
4.7 Quality measures employed in the sensitivity experiment 60
4.8 Variability of specification measures in the context of drift detection. 61
4.9 Aggregate confidence in the Help-Desk sub-logs 64
4.10 Aggregate confidence in the Sepsis sub-logs 66

5.1 Event log 73
5.2 Specification generated from the Hospital Workflow net 74
5.3 Specification generated from the Pharmaceutical company Workflow net 74
5.4 Specification generated from the Specialized clinic Workflow net . . 74
5.5 Specification generated from the Hospital Workflow net 75
5.6 Event logs used for our experiments 83

Chapter 1

Introduction

Modeling behavior is a fundamental activity in domains dealing with the analysis, control, and adaptation of complex systems, including business process management [80], autonomous systems [158], and systems engineering [112]. At its core, behavioral modeling abstracts from concrete executions to provide explicit representations of how systems act, interact, and evolve over time, thus enabling reasoning across different application contexts, ranging from the description of expected executions to the constraint of admissible actions and the guidance of adaptive decision making.

In a business setting, modeling process behavior yields explicit structures of activities, control-flow relations, or execution constraints, aiming to enhance comprehension of the overall operational workflow and to identify and prevent potential issues [113]. This endeavor can be considered a key step throughout an organization's lifespan, as it directly affects how work is performed and, consequently, the generation of value and operational efficiency. As processes evolve over time and execution data become increasingly available, process representations are no longer used solely for design purposes, but also as an analytical basis to reason about observed behavior. This perspective has fostered the development of process mining techniques [13], which aim at extracting, analyzing, and validating process representations based on executions recorded by information systems, i.e., the *event logs* [63]. Within this context, a variety of representation paradigms have been proposed to allow organizations to depict and describe acceptable process behavior. Generally, the so-called *imperative* paradigms (e.g., Workflow nets [4] and BPMN [80]) offer the capabilities to explicitly capture sequences of actions and options available at each stage of the execution. In the literature, imperative process representations are also commonly referred to as *procedural*, and the two terms are often used interchangeably [84]. These representations are particularly effective in capturing well-structured processes with clearly defined control-flow, but their explicit nature may become a

limitation in settings characterized by high variability or loosely structured behavior. In many organizational settings, the same high-level routine remains recognizable while its concrete executions show significant variation across cases, actors, and contextual conditions. This tension between an aspirational view of "how work is supposed to be done" and a realistic view of "how work is actually done" has been extensively discussed in organizational studies, highlighting that stability of intent can coexist with variability of execution [143]. In the above scenarios, enumerating all admissible execution paths through means of imperative representations may lead to large, cluttered (so-called *spaghetti*) models and limit both comprehension and analysis [96]. This motivates the need for alternative representations that capture acceptable behavior without committing to a single prescriptive control-flow structure. *Declarative specifications* (e.g., DECLARE [145] and DCR Graphs [101]) describe process behavior in terms of constraints that executions must satisfy, instead of explicitly detailing all allowed execution paths. Such specifications provide an advantage when operating in intricate and mutable environments, which stems from their effectiveness in describing the behavior in a more flexible way [10]. A declarative specification takes the form of a set of constraints that can be incrementally added, refined, or combined to reflect different requirements, regulations, or stakeholder perspectives, while continuing to support joint reasoning about overall behavior against recorded process execution data. This last operation requires an appropriate semantics, which is made explicit in DECLARE, by interpreting executions as finite sequences of events (i.e., finite traces) and specifying constraints as temporal properties whose satisfaction can be evaluated over such traces. To this end, Linear Temporal Logic on finite traces (LTL_f) constitutes the foundational framework to express temporal properties of finite executions and to evaluate their satisfaction on observed behavior [54]. Accordingly, DECLARE specifications are defined as sets of constraint templates, each of which can be mapped to an LTL_f formula.

The rationale behind this thesis is to employ declarative specifications as an instrument for automated reasoning in process science. In this thesis, thus, we do not consider their extant role as a representation that is alternative to procedural approaches (Workflow nets and the like). A number of papers have already demonstrated its ineffectiveness in terms of comprehensibility to end-users [84]. Instead, we leverage declarative specifications as a behavioral layer that remains "under the hood" to steer and frame automated discovery, diagnostics, conformance checking, and execution of processes.

In light of these considerations, the central research objective pursued in this thesis is articulated as follows:

RO: Declarative Specifications for Process-Aware Systems

Investigate the use of LTL_f declarative specifications as an abstraction layer for representing, verifying, and exploiting behavioral knowledge in process-aware systems.

To follow this objective, we set a series of three research questions covering representation, verification and application of declarative specifications.

RQ1: Process Representation Paradigms

How can a declarative process specification be systematically derived from an imperative process model, preserving behavioral equivalence between the two representations?

Over the years, research has acknowledged that none of the available process representations would be superior in all cases, as imperative and declarative representations are suited to different comprehension tasks [147]. Therefore, seeking to exploit the benefits inherent in each representation, a range of approaches has been proposed to represent a process through a combination of them. This raises the question of which components are most appropriately assigned to each paradigm [174].

Enabling transitions between imperative and declarative representations of the same behavior would allow analysts to navigate across representation paradigms and selectively adopt the most suitable one, adjusting appropriate techniques according to the reasoning or analysis objectives being pursued [11]. The first research to pursue in this direction is that of Prescher et al. [152], which demonstrated the feasibility of moving from a process model in the form of a declarative specification to an equivalent imperative model in the form of a Petri Net. Despite the line of research delineated, there is no direct conversion mechanism that derives a declarative specification that is behaviorally equivalent to an imperative model. To answer RQ1, we introduce a systematic approach to synthesize declarative process specifications in the form of a subset of DECLARE constraints from any input safe and sound Workflow net. We formally prove that the net and the output specification are bisimilar. This result facilitates comprehension and integration between models, not least providing helpful support for behavior verification using process diagnostics tools. We validated the approach through a proof-of-concept implementation, evaluating scalability on synthetic and real-world testbeds and showcasing its applicability.

RQ2: Conformance Measurement

How can the satisfaction of declarative process specifications be quantitatively assessed over observed executions?

The usefulness of discovered process representation, either imperative or declarative, depends on the ability to relate them to observed executions. In process mining, this relation is typically established through *conformance checking* [9], which aims to assess whether recorded behavior complies with a given model. Traditional conformance checking approaches often adopt a qualitative perspective, classifying executions as either conformant or non-conformant. However, when dealing with real-life event logs, such binary assessments may be insufficient to capture the degree, relevance, or stability of observed behavior. This limitation becomes particularly evident in settings characterized by noise, variability, or behavioral changes over time. This motivates the need for quantitative measures that assess conformance of a given set of declarative constraints in a graded manner by means of probabilistic interpretations.

To answer RQ2, we introduce interestingness measures that gauge the degree to which LTL_fconstraints are supported by observed behavior. These measures enable behavioral diagnostics and provide a baseline for detecting deviations, changes, and emerging patterns in process executions.

> **RQ3: Decentralized Data Sources**
>
> How can declarative specifications be checked against execution data that is distributed across multiple independent sources?

In process-oriented settings, behavior is frequently observed across multiple independent data sources, each retaining control over its own execution data, rules, and constraints. While the availability of such distributed information enables richer behavioral analysis, it also introduces fundamental challenges related to data secrecy, trust, and governance. In particular, the aggregation and analysis of execution data may reveal sensitive information that data providers are unwilling or unable to disclose to external parties. In these inter-organizational settings, behavioral requirements are typically not governed by a single regulation. Each party may impose its own compliance obligations, privacy constraints, and operational policies. Declarative specifications fit this setting as they are modular sets of constraints that can be checked locally against the behavior visible to each organization and, when permitted, composed into an overall specification for joint analysis. Companies, though, are reluctant to share private information required to execute process mining and conformance checking algorithms with external parties [127], thus hindering their adoption. Letting sensitive operational data traverse organizational boundaries introduces concerns about data secrecy, security, and compliance with internal regulations [138]. To meet the RQ3, we propose CONFINE, a novel approach and tool aimed at enhancing collaborative information system architectures with

secrecy-preserving process mining capabilities. To secure information secrecy during the exchange and elaboration of data, our solution resorts to *Trusted Execution Environments* (TEEs) [159], namely hardware-secured contexts that guarantee code integrity and data confidentiality before, during, and after their utilization. Owing to these characteristics, CONFINE lets information be securely transferred beyond the organization's borders. Therefore, computing nodes other than the information provisioners can aggregate and elaborate the original, unaltered process data in a secure, externally inaccessible vault.

The remainder of the thesis is organized as follows. Chapter 2 introduces the background concepts and formalisms that frame the contributions of this work. Chapter 3 addresses the first research question (RQ1), formally defining how declarative specifications can be systematically derived from imperative representations. Chapter 4 focuses on the analysis of declarative process specifications with respect to observed executions, introducing quantitative conformance metrics, thus answering the second research question (RQ2). Chapter 5 focuses on the third research question (RQ3), extending the conformance checking application to decentralized scenarios, in which behavioral data are distributed across different data sources. Finally, Chapter 6 concludes the thesis, summarizing the findings in light of the research objective (RO), and outlines directions for future research.

Chapter 2

Background

This chapter introduces the background concepts and the main formalisms that provide the foundations for the contributions presented in the remainder of the thesis.

Section 2.1.1 introduces the main formalism of the *imperative* process representations, including Petri nets and Workflow nets. Section 2.1.2 introduces Linear Temporal Logic on finite traces (LTL_f) as a formal language to specify behavioral constraints over executions, which in turn supports the definition of *declarative* process specifications (e.g., DECLARE). Finally, Section 2.2 discusses event logs and behavioral verification, reviewing the core notions and the approaches to conformance checking.

2.1 Process Representations and Semantics

2.1.1 Petri Nets and Workflow Nets

Petri nets are a classical formalism for modeling distributed and concurrent systems, originally introduced by Carl Adam Petri to capture causality, synchronization, and resource sharing [146]. Their application in process-aware information systems stems from the fact that they provide both an easy-to-understand graphical notation and a mathematically based execution semantics, making them suitable for representing control-flow and formal verification [64].

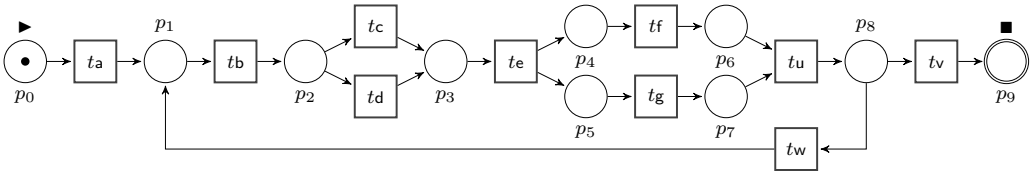


Figure 2.1. A Workflow net

Definition 1 (Petri net). *A place/transition net [64] (henceforth,* Petri net*) is a bipartite graph $\mathcal{PN} = (P, T, F)$, where P (the finite set of "*places*") and T (the finite set of "*transitions*") constitute the nodes ($P \cap T = \emptyset$), and the* flow relation *$F \subseteq (P \times T \uplus T \times P)$ defines the edges.* ◁

In the following, we shall name "*underlying graph*" the graph obtained from a Petri net $\mathcal{PN} = (P, T, F)$ whereby the set of nodes is $P \cup T$ and the directed edges' relation is F. Given a place $p \in P$, we shall denote the sets $\{t \mid (t, p) \in F\}$ and $\{t \mid (p, t) \in F\}$ with $\bullet p$ ("*preset*") and $p\bullet$ ("*postset*"), respectively. For example, in Fig. 2.1, $\bullet p_3 = \{\mathsf{tc}\}$ and $p_3\bullet = \{\mathsf{te}, \mathsf{tf}\}$.

Within Business Process Management (BPM), a *Workflow net* (see, e.g., Fig. 2.1) is a well-known subclass of Petri nets used to formally represent imperative process models [4]. Workflow nets tailor Petri nets to the life-cycle of a single process instance by requiring a unique *initial place* and a unique *output place*, and by ensuring that every node lies on a path from the initial to the output place. These structural constraints align the model with the notion of a process that starts, progresses through activities, and eventually completes, which makes Workflow nets a baseline for modeling business processes.

Definition 2 (Workflow net). *A Workflow net $\mathcal{WN} = (P, T, F)$ is a Petri net such that:*

1. *There is a unique place ("*initial place*", $\blacktriangleright \in P$) such that its preset is empty;*
2. *There is a unique place ("*output place*", $\blacksquare \in P$) such that its postset is empty;*
3. *Every place $p \in P$ and transition $t \in T$ is on a path of the underlying graph from $\blacktriangleright$ to $\blacksquare$.* ◁

We remark that we operate with full knowledge of the imperative model's structure, treated as a white box. Therefore, we directly focus on transitions rather than on their labels here.

In Petri and Workflow nets, places can be *marked* with tokens, intuitively representing resources that are processed by the transitions succeeding them in the net. In Fig. 2.1, a token is graphically depicted as a solid circle (see p_0 in the figure). The state of a net is defined by the distribution of tokens over places. This is formalized with the notion of *marking*, a function mapping each place to the number of tokens in it. The net's state changes with the consumption and production of tokens caused by the execution ("*firing*") of transitions.

Definition 3 (Marking and firing). *Let $\mathcal{WN} = (P, T, F)$ be a Workflow net. A* marking *is a function $M : P \to \mathbb{N} \cup \{0\}$. The* initial marking *$M_0$ of $\mathcal{WN}$ maps $\blacktriangleright$ to 1 and any other $p \in P \setminus \{\blacktriangleright\}$ to 0. A marking M of $\mathcal{WN}$ is* final *if $M(\blacksquare) > 0$. A*

marking enables *a transition* $t \in T$ *iff* $M(p) > 0$ *for all places* p *such that* $t \in p\bullet$. *An enabled transition can* fire, *i.e., turn a marking* M *into* M' *(in symbols,* $M[t\rangle M'$*), according to the following rule: For each place* $p \in P$, $M'(p) = M(p) + 1$ *if* $t \in \bullet p$; $M'(p) = M(p) - 1$ *if* $t \in \bullet p$; *otherwise,* $M'(p) = M(p)$. ◁

In Fig. 2.1, e.g., the initial marking enables $t\mathsf{a}$. Denoting markings with a multi-set notation, $\{p_0\}\,[t\mathsf{a}\rangle\,\{p_1\}$. Subsequently, $t\mathsf{b}$ gets enabled. After firing $t\mathsf{b}$, and $t\mathsf{c}$ get enabled. With Petri and Workflow nets, interleaving semantics are adopted, thus only one transition can fire per timestep, thus the firing of $t\mathsf{b}$ and $t\mathsf{c}$ are mutually exclusive in that state.

Definition 4 (Firing sequence and run). *Given a Workflow net* $\mathcal{WN} = (P, T, F)$, *a (finite)* firing sequence σ *is* $\langle\rangle$ *or a sequence of transitions* $\langle t_1, \ldots, t_n\rangle$ *such that, for any index* $1 \leq i \leq n$ *with* $n \in \mathbb{N}$, $t_i \in T$: *(a) The* i*-th marking enables the* i*-th transition; (b) The* $i+1$*-th marking* M_{i+1} *is such that* $M_i[t_i\rangle M_{i+1}$. *A marking* M' *is* reachable *in* $\mathcal{WN}$ *if there exists a firing sequence* σ *leading from the initial marking* M_0 *to* M' *(in symbols,* $M_0[\sigma\rangle M'$*). A firing sequence leading from* M_0 *to a final marking is a* run. ◁

Given a workflow net $\mathcal{WN}$, we will use $\mathcal{M}_{\mathcal{WN}}$ to denote the set of markings that can be reached from its initial marking M_0.

Runs of the Workflow net in Fig. 2.1 include $\langle t\mathsf{a}, t\mathsf{b}, t\mathsf{c}, t\mathsf{e}, t\mathsf{f}, t\mathsf{g}, t\mathsf{u}, t\mathsf{v}\rangle$ and $\langle t\mathsf{a}, t\mathsf{b}, t\mathsf{d}, t\mathsf{e}, t\mathsf{f}, t\mathsf{g}, t\mathsf{u}, t\mathsf{w}, t\mathsf{c}, t\mathsf{e}, t\mathsf{f}, t\mathsf{g}, t\mathsf{u}, t\mathsf{v}\rangle$. Any prefix of the first run of length 7 or less is a firing sequence from the initial marking $\{p_0\}$ but not a run.

In this paper, we assume that Workflow nets enjoy the following properties.

Definition 5 (Soundness and safety of a Workflow net). *Let* $\mathcal{WN} = (P, T, F)$ *be a Workflow net.* $\mathcal{WN}$*is* k-bounded *if the number of tokens assigned by any reachable marking* M' *to any place* $p \in P$ *is such that* $M'(p) \leq k$. *A 1-bounded Workflow net is* safe. $\mathcal{WN}$*is* sound *iff it enjoys the following properties:* **Option to complete:** *from any marking* M *it is possible to reach the final marking;* **Proper completion:** *if a reachable marking* M *is such that* $M(\blacksquare) > 0$, *then* M *is the final marking;* **No dead transitions:** *for any transition* $t \in T$, *there exists a reachable marking* M *such that* t *is enabled by* M.

Safe and sound Workflow nets (like the one depicted in Fig. 2.1) are a superclass of sound S-coverable nets, which in turn subsume safe and sound free-choice and well-structured nets [2]. These structural characteristics are widely recognized as recommendable in process management [4] and underpin well-formed business process diagrams [142]. Notice that, given a *safe* net, all markings that are reachable from the initial one are such that each place can be marked with at most one token. Also, the final marking of a sound workflow net is $\{\blacksquare\}$.

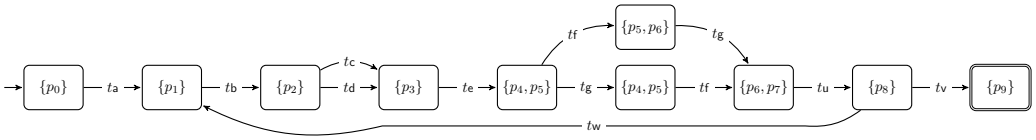


Figure 2.2. Reachability graph derived from the Workflow net in Fig. 2.1.

The state space of k-bounded Petri nets can be represented in the form of a deterministic labeled transition system that go under the name of *reachability graph* [64]. Safe and sound Workflow nets have a given initial marking and one final marking. We can thus endow the state representation with these characteristics and reinterpret the known concept of reachability graph as a finite state automaton.

Definition 6 (Finite State Automaton). *A (deterministic) finite state automaton (FSA) is a tuple $\mathcal{A} = (S, s_0, s_F, \Sigma, \delta)$, where S is a finite set of states, $s_0 \in S$ is the initial state, $s_F \subseteq S$ is the set of accepting states, Σ is the input alphabet of the automaton, and $\delta : S \times \Sigma \to S$ is the state transition function.* ◁

An finite state automaton (FSA) reads in input sequences of symbols ("*string*") of its input alphabet. To formally define how an automaton processes an input string, we extend the transition function δ to strings through its inductive extension $\widehat{\delta} : S \times \Sigma^* \to S$ defined as follows:

$$\widehat{\delta}(s, \epsilon) = s$$
$$\widehat{\delta}(s, wa) = \delta(\widehat{\delta}(s, w), a)$$

where ϵ denotes the empty string, $w \in \Sigma^*$ is a string, and $a \in \Sigma$ is a symbol.

A string $w \in \Sigma^*$ is accepted by $\mathcal{A}$ iff $\widehat{\delta}(s_0, w) \in s_{\mathrm{F}}$. The language accepted by $\mathcal{A}$ is therefore:

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \widehat{\delta}(s_0, w) \in s_{\mathrm{F}}\}.$$

Intuitively, the automaton starts from the initial state and updates its current state according to the transition function while reading the input string symbol by symbol.

Definition 7 (Bisimilarity). *Two FSAs $\mathcal{A} = (S, s_0, s_F, \Sigma, \delta)$ and $\mathcal{A}' = (S', s'_0, s'_F, \Sigma, \delta')$ are* bisimilar *if and only if there exists a relation $\sim \subset S \times S'$ such that the following hold: $(s_0, s'_0) \in \sim$; if $(s, s') \in \sim$, then $(\delta(s, \ell), \delta'(s', \ell)) \in \sim$ for any $\ell \in \Sigma$; if $(s, s') \in \sim$, then $s \in s_F$ if and only if $s' \in s'_F$.* ◁

Observation 1. *In the case of FSAs, bisimilarity coincides with* language equivalence*, i.e., two FSAs are bisimilar if and only if the sets of strings that they accept are equal [103].* ◁

Definition 8 (Reachability FSA). *Given a sound and safe Workflow net $\mathcal{WN}$, the reachability FSA $\mathcal{A}^{\mathcal{WN}} = \left(S^{\mathcal{WN}}, s_0^{\mathcal{WN}}, s_F^{\mathcal{WN}}, \Sigma^{\mathcal{WN}}, \delta^{\mathcal{WN}}\right)$ is a finite state automaton where:*
$S^{\mathcal{WN}} = \mathcal{M}_{\mathcal{WN}}$ *, i.e., the set of states is the set of reachable markings in $\mathcal{WN}$;*
$s_0^{\mathcal{WN}} = \{\blacktriangleright\}$ *, i.e., the initial state is the initial marking of $\mathcal{WN}$;*
$s_{\mathbf{F}}^{\mathcal{WN}} = \{\{\blacksquare\}\}$ *, i.e., the accepting state set is a singleton containing the final marking of $\mathcal{WN}$;*
$\Sigma^{\mathcal{WN}} = T$ *, i.e., the alphabet is the set of transitions of $\mathcal{WN}$;*
$\delta^{\mathcal{WN}}$ *is s.t. $\delta(M, t) = M'$ iff $M[t\rangle M'$ for every transition t and reachable M, M' in $\mathcal{WN}$.* ◁

Figure 2.2 depicts the reachability FSA of the Workflow net in Fig. 2.1.

Observation 2. *The language of a safe and sound Workflow net is regular. Indeed, safety guarantees that the number of reachable markings is finite, while soundness provides a unique accepting marking. Therefore, the reachability graph of a safe and sound Workflow net can be interpreted as a finite state automaton (Def. 8) recognizing exactly the runs of the Workflow net.* ◁

When dealing with accepting traces and languages, working with trimmed or non-trimmed FSAs is equivalent. This is not the case when we structurally relate the FSA of a DECLARE specification with the reachability FSA of a Workflow net. That FSA is indeed constructed, state-by-state, considering only enabled transitions, which globally yields that it is trimmed by design. Therefore we operate with trimmed FSAs for this comparison.

2.1.2 Linear Temporal Logic on finite traces LTL$_f$

LTL$_f$ has the same syntax as LTL [149], but is interpreted on finite traces and is at the basis of declarative process specification languages such as DECLARE [71]. Here, we consider the LTL dialect including past modalities [126]. From now on, we fix a finite set Σ representing an alphabet of propositional symbols. A (finite) *trace* $\pi = \langle a_1, \ldots, a_n \rangle \in \Sigma$ is a finite sequence of symbols of length $|\pi| = n$ (with $n \in \mathbb{N}$), where the occurrence of symbol a_i at instant i of the trace represents an *event* that witnesses a_i at instant i —we write $\pi(i) = a_i$. Notice that *at each instant we assume that one and only one symbol occurs.* Using standard notation from regular expressions, Σ^* denotes the overall set of finite traces derived from events belonging to Σ.

Definition 9 (Syntax of LTL$_f$). *Well-formed Linear Temporal Logic on Finite Traces (LTL$_f$) formulae are built from an alphabet $\Sigma \supseteq \{\mathrm{a}\}$ of propositional symbols,*

auxiliary symbols '(' and ')', propositional constants True *and* False*, the logical connectives* $\neg$ *and* $\wedge$*, the unary temporal operators* $\bigcirc$ *(*next*) and* $\ominus$ *(*yesterday*), and the binary temporal operators* $\mathbf{U}$ *(*until*) and* $\mathbf{S}$ *(*since*) as follows:*

$$\varphi ::= True | False | a | (\neg\varphi) | (\varphi_1 \wedge \varphi_2) | (\bigcirc\varphi) | (\varphi_1 \ \mathbf{U} \ \varphi_2) | (\ominus\varphi) | (\varphi_1 \ \mathbf{S} \ \varphi_2). \qquad \triangleleft$$

We may omit parentheses when the operator precedence intuitively follows from the expression. Given $\{\mathsf{e}, \mathsf{d}\} \subseteq \Sigma$, e.g., the following is an LTL_f formula: $(\bigcirc\neg\mathsf{e}) \ \mathbf{U} \ \mathsf{d}$.

Semantics of LTL_f is given in terms of finite traces, i.e., finite words over the alphabet 2^Σ. We name the index of the element in the trace as *instant*. Intuitively, $\bigcirc\varphi$ and $\ominus\varphi$ indicate that φ holds true at the next and previous instant, respectively; $\varphi_1 \ \mathbf{U} \ \varphi_2$ states that φ_2 will eventually hold and, until then, φ_1 holds too; dually, $\varphi_1 \ \mathbf{S} \ \varphi_2$ signifies that φ_2 holds at some point and, from that instant on, φ_1 holds too. We formalize the above as follows.

Definition 10 (Semantics of LTL_f). *Given a finite trace* π *of length* $n \in \mathbb{N}$*, an* LTL_f *formula* φ *is satisfied at a given instant* i *(*$1 \leq i \leq n$*) by induction of the following:*

$(\pi, i) \models$ *True;* $(\pi, i) \not\models$ *False;*

$(\pi, i) \models a$ *iff* a *is True in* $\pi(i)$*;*

$(\pi, i) \models \neg\varphi$ *iff* $(\pi, i) \not\models \varphi$*;*

$(\pi, i) \models \varphi_1 \wedge \varphi_2$ *iff* $(\pi, i) \models \varphi_1$ *and* $(\pi, i) \models \varphi_2$*;*

$(\pi, i) \models \bigcirc\varphi$ *iff* $i < n$ *and* $(\pi, i+1) \models \varphi$*;*

$(\pi, i) \models \ominus\varphi$ *iff* $i > 1$ *and* $(\pi, i-1) \models \varphi$*;*

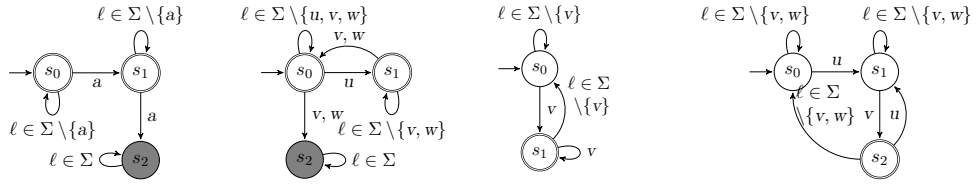
$(\pi, i) \models \varphi_1 \ \mathbf{U} \ \varphi_2$ *iff* $(\pi, j) \models \varphi_2$ *with* $i \leq j \leq n$*, and* $(\pi, k) \models \varphi_1$ *for all* k *s.t.* $i \leq k < j$*;*

$(\pi, i) \models \varphi_1 \ \mathbf{S} \ \varphi_2$ *iff* $(\pi, j) \models \varphi_2$ *with* $1 \leq j \leq i$*, and* $(\pi, k) \models \varphi_1$ *for all* k *s.t.* $j < k \leq i$*.* $\triangleleft$

Without loss of generality, we consider here the non-strict semantics of $\mathbf{U}$ and $\mathbf{S}$ [102]. In the following, we might directly refer to the sequence of events $\langle e_1, \ldots, e_n \rangle$ of a trace π of length n to indicate the sequence of assignments at instants $1, \ldots, n$.

From the above operators, the following can be derived:

- Classical boolean abbreviations $\vee, \rightarrow$;
- Constant $\pi_{\text{End}} \equiv \neg \bigcirc True$, the last instant of a trace;
- Constant $\pi_{\text{Start}} \equiv \neg \ominus True$, the first instant of a trace;
- $\Diamond\varphi \equiv True \ \mathbf{U} \ \varphi$, indicating that φ holds true at an instant between the current one (included) and π_{End} (we name this operator *eventually*);
- $\Diamond\!\!\!-\varphi \equiv True \ \mathbf{S} \ \varphi$, indicating that φ holds true at an instant in the closed interval from π_{Start} to the current one (*once*);



(a) AtMostOne (a) **(b)** Alt.Prec. $(u, \{v, w\})$ **(c)** End (v) **(d)** $\{$Alt.Prec. $(u, \{v, w\})$, End $(v)\}$

Figure 2.3. Example FSAs of Declare constraints. The FSA in Fig. 2.3(d) is trimmed.

Table 2.1. Semantics of some Declare constraint templates

Template	LTL_f expression [58, 47]	Description
AtMostOne (x)	$\Box(x \rightarrow \neg \bigcirc \Diamond x)$	x occurs at most once in the trace
End (x)	$\Box \Diamond x$	The last event of any trace is x
AlternatePrecedence (y, x)	$\Box(x \rightarrow \ominus(\neg x \ \mathbf{S} \ y))$	Every occurrence of x requires that y occurred before, with no recurrence of x in between

- $\Box\varphi \equiv \neg\Diamond\neg\varphi$, indicating that φ holds true at every instant from the current one till π_{End} (*always*);
- $\boxminus\varphi \equiv \neg\blacklozenge\neg\varphi$, indicating that φ holds true at every instant from π_{Start} to the current one (*historically*).

For example, $\mathsf{d} \wedge \Diamond \mathsf{e}$ is satisfied in a trace when the propositional atom d holds true and e holds true at a later instant in the same trace.

Notice that the semantics of the *past operators* $\ominus$, $\mathbf{S}$, $\blacklozenge$, and $\boxminus$ correspond to the semantics of the *future operators* $\bigcirc$, $\mathbf{U}$, $\Diamond$, and $\Box$, respectively, if we evaluate them on finite traces read in reverse [45].

Let $\|\varphi\|$ denote the size of the LTL_f formula φ in terms of propositional symbols and connectives excluding parentheses. For example, $\|(\bigcirc\neg\mathsf{e}) \ \mathbf{U} \ \mathsf{d}\|$ is 5 and $\|\mathsf{d} \wedge \Diamond\mathsf{e}\|$ is 4.

Every LTL_f formula can be encoded into a *deterministic finite state automaton* [55]. Figure 2.3 depicts three FSAs. A direct approach that builds a nondeterministic FSA $\mathcal{A}_\varphi$ accepting all and only the traces that satisfy a given LTL_f formula φ is presented in [55]. We make two further observations from [103]: *(i)* the so-obtained FSAs can be determinized, minimized, and trimmed (i.e., all unreachable states and states from which no accepting state can be reached are removed) using standard techniques without modifying the accepted language, and *(ii)* given any two FSAs $\mathcal{A}_\varphi$ and $\mathcal{A}_{\varphi'}$, their *product* $\mathcal{A}_\varphi \times \mathcal{A}_{\varphi'}$ recognizes all and only the traces of $\varphi \wedge \varphi'$.

2.1.3 Declarative Specifications

The declarative specification [145] of a process is a representation language that captures expected behavior through a set of constraints, rather than by explicitly

defining all allowed execution paths. Its building blocks are constraint patterns that can be instantiated over activities (or, more generally, propositional symbols Σ) to express temporal relations. In this work, we resort on the DECLARE templates repertoire. The semantics of a DECLARE template is given as an LTL_f formula, so that checking a DECLARE specification reduces to evaluating the corresponding temporal constraints over execution traces. Given the free variables ("*parameters*") x and y, e.g., ALTERNATEPRECEDENCE (y, x) corresponds to $\Box(x \rightarrow \ominus(\neg x \ \mathbf{S} \ y))$, witnessing that for every instant in which x is verified, then a previous instant must verify y without any occurrences of x in between. Hitherto, we will occasionally use an abbreviation for the template name i.e., ALT.PREC. (y, x). Table 2.1 shows the LTL_f formulae of some templates of the DECLARE repertoire. Standard DECLARE imposes that template parameters be interpreted as single symbols of Σ to build *constraints*. For example, ALT.PREC. (b, c) interprets x as c and y as b.

Branched DECLARE [145] comprises the same set of templates of standard DECLARE, yet allowing the interpretation of parameters as elements of a join-semilattice $(\Sigma, \vee)$, i.e., an idempotent commutative semigroup, where $\vee$ is the join-operation [67]. We shall use a clausal set-notation whenever a parameter is interpreted as a disjunction of literals. For example, ALT.PREC. $(\{a, w\}, b)$ interprets x as b and y as $a \vee w$: for every b occurring in a trace, a previous instant must have verified $a \vee w$, without b recurring between that instant and the following occurrence of b. The conjunction of a finite set of constraints forms a DECLARE specification. In the following, we formalize the above notions.

Definition 11. *A DECLARE specification $\mathcal{DS} = (\text{REP}, \Sigma, K)$ is a tuple wherein:*

REP is a finite non-empty set of templates*, or "*repertoire*", where each template $\kappa(x_1, \ldots, x_m)$ is an LTL_f formula parameterized on free variables $x_1, \ldots, x_m$;*

$\Sigma \ni \mathbf{a}_i$ is a finite non-empty alphabet of symbols a_i *with $1 \leq i \leq |\Sigma|$, $|\Sigma| \in \mathbb{N}$;*

K is a finite set of constraints*, namely pairs $(\kappa(x_1, \ldots, x_m), \kappa)$ where $\kappa(x_1, \ldots, x_m)$ is a template from REP, and $\kappa : \{x_1, \ldots, x_m\} \rightarrow 2^{\Sigma} \setminus \{\}$ is a mapping from every variable x_i to a non-empty, finite set of symbols $A_i = \{\mathsf{a}_{i,1}, \ldots, \mathsf{a}_{i,v_i}\} \subseteq \Sigma$, with $1 \leq i \leq m$ and $1 \leq v_i \leq |\Sigma|$; we denote such a constraint with $\kappa(A_1, \ldots, A_m)$ or equivalently $\kappa(\{\mathsf{a}_{1,1}, \ldots, \mathsf{a}_{1,v_1}\}, \ldots, \{\mathsf{a}_{m,1}, \ldots, \mathsf{a}_{m,v_m}\})$, omitting curly brackets from the latter form whenever a variable is mapped to a singleton.* ◁

As an example, consider the following specification: REP $= \{$ATMOSTONE (x), END (x), ALT.PREC. $(y, x)\}$, $\Sigma = \{a, b, c, d, e, f, g, u, v, w\}$, and $K = \{$ATMOSTONE (a), END (v), ALT.PREC. (e, f), ALT.PREC. $(\{a, w\}, b)$, ALT.PREC. $(u, \{v, w\})\}$, where, e.g., ALT.PREC. $(\{a, w\}, b)\}$ is derived from the template ALT.PREC. (y, x) by mapping $y \mapsto_{\kappa} \{a, w\}$ and $x \mapsto_{\kappa} b$.

Definition 12 (Constraint formula, satisfying trace). *Let* $\kappa(A_1, \ldots, A_m)$ *be a constraint, whereby* $A_i = \{a_{i,1}, \ldots, a_{i,v_i}\}$ *for each* $1 \leq i \leq m$. *Its* constraint formula, *written* $\varphi_{\kappa(A_1,\ldots,A_m)}$, *is the* LTL_f *formula obtained from the template* $\kappa(x_1, \ldots, x_m)$ *by interpreting* x_i *as* $(a_{i,1} \vee \cdots \vee a_{i,v_i})$ *for each* $1 \leq i \leq m$. *A trace* π satisfies $\kappa(A_1, \ldots, A_m)$ *iff* $\pi \models \varphi_{\kappa(A_1,\ldots,A_m)}$; *otherwise, we say that* π violates $\kappa(A_1, \ldots, A_m)$. ◁

Considering Tab. 2.1 and the above example specification, we have that $\varphi_{\text{Alt.Prec.}(\{a,w\},b)} = \Box(b \rightarrow \ominus(\neg b \ \mathbf{S} \ (a \vee w)))$, and $\varphi_{\text{End}(v)} = \Box \Diamond v$. Traces $\langle a, b, c\rangle$, $\langle a, b, c, f, u, w, b\rangle$, and $\langle a, b, c, e, f, g, u, v\rangle$ satisfy Alt.Prec. $(\{a, w\}, b)$, while only the third one satisfies End (v).

Definition 13 (Specification formula, model trace). *A given* Declare *specification* $\mathcal{DS} = (Rep, \Sigma, K)$ *is logically represented by conjoining its constraint formulae* $\varphi_{\mathcal{DS}} \doteq \bigwedge_{\kappa(A_1,\ldots,A_m)\in K} \left(\varphi_{\kappa(A_1,\ldots,A_m)}\right)$. *A trace is a* model trace *for the specification,* $\pi \models \mathcal{DS}$, *iff* $\pi \models \varphi_{\mathcal{DS}}$, *i.e., it satisfies the conjunction of all the constraint formulae,* $\pi \models \varphi_{\kappa(A_1,\ldots,A_m)}$ *for each* $\kappa(A_1, \ldots, A_m) \in K$. ◁

The specification formula of the above example is:

$$\begin{aligned}
&\big(\Box(a \rightarrow \neg \bigcirc \Diamond a)\big) \wedge \big(\Box \Diamond v\big) \wedge \\
&\big(\Box(f \rightarrow \ominus(\neg f \ \mathbf{S} \ e))\big) \wedge \\
&\big(\Box(b \rightarrow \ominus(\neg b \ \mathbf{S} \ (a \vee w)))\big) \wedge \\
&\big(\Box((v \vee w) \rightarrow \ominus(\neg(v \vee w) \ \mathbf{S} \ u))\big).
\end{aligned}$$

$\langle a, b, c, e, f, g, u, v\rangle$ is a model trace for it, unlike $\langle a, b, c\rangle$ or $\langle a, b, c, f, u, w, b\rangle$.

Leveraging the techniques mentioned at the end of Def. 6 and the above definition, we can create an FSA that accepts all and only the traces of a single Declare formula φ and of a whole specification $\mathcal{DS}$.

Definition 14 (Constraint and specification FSA). *Let* $\varphi_1, \ldots, \varphi_{|K|}$ *be the constraint formulae of a process specification* $\mathcal{DS}$. *A* constraint automaton $\mathcal{A}_{\varphi_i}$ *is an FSA that accepts all and only those traces that satisfy* φ_i *[68] with* $1 \leq i \leq |K|$. *The product automaton* $\mathcal{A}_{\varphi_1} \times \cdots \times \mathcal{A}_{\varphi_{|K|}}$ *is the* specification FSA, *recognizing all and only the traces satisfying* $\mathcal{DS}$. ◁

Figures 2.3(a) to 2.3(c) show the automata of constraints AtMostOne (a), Alt.Prec. $(u, \{v, w\})$, and End (v), respectively. Figure 2.3(d) depicts the FSA of a specification consisting of End (v) and Alt.Prec. $(u, \{v, w\})$. Notice that the accepting state cannot be reached from s_2 in Figs. 2.3(a) and 2.3(b). Instead, the FSA in Fig. 2.3(d) has no such trap states due to trimming.

Aside from keeping the FSA's language unchanged, trimming caters for structural compatibility with the state space representation of Workflow nets, which we discuss next.

2.2 Event Logs and Behavior Verification

In the followings we introduce the data and verification perspective adopted throughout the rest of the thesis. We first formalize *event logs* as collections of recorded executions, which provide the observational basis for process analysis. We then recall core *process mining* tasks, focusing on *conformance checking*, which is the procedure of assessing whether observed executions comply with a given process representation. This lays the groundwork for the quantitative analyses presented in the remainder of this thesis.

2.2.1 Event Logs

In information systems, process executions are increasingly recorded as digital footprints. As process instances progress through activities, systems log the corresponding sequence of *events*, often enriching each event with attributes such as the executed activity, timestamps, and the involved human or system resources. This growing availability of execution data provides an observational perspective to the process representations introduced in Sect. 2.1, enabling empirical analysis of how processes actually run with respect to the representation. Since the same process can be executed many times, we formalize the recorded data as an *event log*, i.e., a multi-set of traces. Events referring to the same process instance are grouped into *cases*, yielding complete *traces* that represent individual executions. Such collections of traces are commonly stored in standardized formats, most notably the *eXtensible Event Stream* (XES) standard [1], which defines a schema for event-level attributes and metadata. *Event logs* will serve as the basis for checking process representations against observed executions.

Definition 15 (Event Log). *Given a finite alphabet of propositional symbols* Σ*, we name as* event *an assignment for the symbols in* Σ *and as* trace *a finite sequence of events. An* event log *(or* log *for short) is a finite multi-set of traces* $L = \{\pi_1^{j_1}, \ldots, \pi_m^{j_m}\}$ *of cardinality* $|L| = \sum_{i=1}^{m} j_i$. ◁

2.2.2 Process Mining

Process mining is a research field at the intersection of data science and process management that aims to analyze and improve real-world process executions based

on event logs recorded by information systems [13, 63]. By leveraging the traces generated during process execution, process mining enables organizations to gain transparency into their operational workflows, verify compliance with prescribed procedures, assess performance, and identify deviations or inefficiencies.

The process mining literature traditionally distinguishes three main families of techniques [6]. *Process discovery*, the probably most common process mining function, aims to mine a process representation that explains the behavioral patterns observed in the recorded executions without a priori knowledge [173, 79, 160]. In the imperative setting, classic discovery algorithms such as the *α-miner* and the *Heuristics Miner* infer control-flow relations by analyzing ordering and dependency patterns between events, producing models (e.g., Petri nets or BPMN diagrams) that can be inspected and analyzed [79]. These representations, as illustrated in Sect. 2.1.1, support visualization of the behavior and provide a basis for subsequent analysis and verification techniques. Complementarily, declarative discovery techniques infer a specification in terms of constraints. In particular, mining tools like MINERful [105] and Janus [47] extract sets of declarative constraints (see Sect. 2.1.2) that capture the observed behavior through temporal rules, enabling verification and diagnostics in flexible settings.

Conformance checking involves the comparison of event data generated during the execution of a process with process representations that define its normative or descriptive behavior [7]. *Enhancement*, which includes all functions that enhance either a process representation or an event log [6, 176].

2.2.3 Conformance Checking

Conformance checking assesses to what extent the observed behavior of a system complies with a given behavioral representation [7]. In process-aware systems, this is typically framed as the comparison between a normative process representation (describing intended behavior) and an event log recording the executions that actually occurred (Def. 15). Beyond detecting deviations, conformance checking aims to quantify compliance and support diagnostics by localizing where and how executions diverge from the reference model. A broad discussion of conformance dimensions and measures is provided in [9]. For imperative models, conformance is commonly addressed by establishing an explicit correspondence between events in a trace and steps in a model execution (often called *replay* or *alignment*), which enables both diagnostics and quantitative scores. For instance, in [157], the authors propose a token-based replay for Petri net models. Here, each trace is replayed from the initial marking, and whenever the next recorded event is not enabled, the procedure records a deviation. Deviations can be quantified by counting missing tokens that must

be added to enable required moves and remaining tokens left behind after replay, indicating improper completion. Although these techniques originated for imperative representations, the same verification need arises for declarative specifications [60]. Here, the reference representation is a set of constraints (e.g., DECLARE), and conformance reduces to checking whether execution traces satisfy the specification, i.e., whether each trace fulfills the required temporal constraints (and, dually, which constraints are violated). This shift from replaying explicit control-flow to evaluating constraint satisfaction is particularly suited to flexible processes, and it supports the quantitative analyses developed in Chapter 4.

Chapter 3

From Imperative Process Models to LTL$_f$ Declarative Specifications

3.1 Introduction

The act of modeling a process is a key element in a multitude of domains, including business process management [80], and is specifically tailored to meet the specific requirements and objectives of the individual application scenarios. Two fundamental, complementary paradigms cover the spectrum of modeling: the imperative (e.g., Workflow nets [4] and BPMN models [80]) and the declarative (e.g., DECLARE maps [145] and DCR Graphs [101]). Generally, the former class offers the opportunity to explicitly capture the set of actions available at each reachable state of the process, from start to end. However, such models often show limitations when it comes to capturing flexibility in execution, since the possible runs highly vary and their graph-based structure gets cluttered. To compactly represent that variability, declarative specifications depict the rules that govern the behavior of every instance, leaving the allowed sequences implicit as long as none of those rules is violated.

Research has acknowledged that none of the available representations would be superior in all cases, as imperative and declarative approaches are apt to different comprehension tasks [147]. The ability to translate one representation to the other while preserving behavioral equivalence would allow the comparison and selection of the most suitable one. The first work in this direction is [152], where a systematic procedure is proposed to turn a declarative specification into an imperative model. Other endeavors followed to close the circle by providing an approximate solution to the inverse path (i.e., from an imperative model to a declarative specification),

resorting to re-discovery over simulations [153], state space exploration [155], or behavioral comparison [32].

In this chapter, we present the solutions to tackle this challenge and to answer the RQ1:

> **RQ1: Process Representation Paradigms**
>
> How can a declarative process specification be systematically derived from an imperative process model, preserving behavioral equivalence between the two representations?

Our goal is to close the existing gap of this procedural-to-declarative direction. To this end, we show how to encode a safe and sound Workflow net [4] into a behaviorally equivalent DECLARE specification [72]. We resorted to only three parametric constraint types (*templates*) in the DECLARE repertoire (hereafter, *"the three spells"*). Importantly, the encoding is obtained in one pass and modularly over the net, preserving runs and choice points without incurring the state space explosion caused by concurrency unfolding. A byproduct of the encoding is that a safe and sound Workflow net induces a star-free regular language when considering transitions of the former as the alphabet of the latter. This is a stronger characterization than regularity, since star-free languages form a strict subclass of regular languages and correspond to first-order definable behaviors over finite traces. The result naturally follows from the fact that the generated DECLARE specification is expressed in LTL$_f$, whose expressive power coincides with first-order logic over finite traces. This strengthens the observation that languages induced by safe and sound Workflow nets are regular.

Then, we evaluate the scalability of our approach by experimentally testing our proof-of-concept implementation against synthetic and real-world testbeds. Also, we show a reasoning task on process diagnostics with public benchmarks.

3.2 Related Work

The relationship between imperative and declarative modeling approaches has been extensively explored in the existing literature, with a prevailing focus directed toward the development of analytics tools that effectively compare and integrate the strengths of both paradigms. Building on previous contributions aimed at establishing a formal connection between these paradigms [152, 51], our research focuses on providing a systematic approach for translating safe and sound Workflow nets into their declarative counterparts. A growing research stream configures this transformation aiming to leverage the support provided by the declarative specifications

for conformance checking and anomaly detection. Notably, integrating declarative constraints into event log analysis facilitates more comprehensive diagnostics than those provided by trace replaying techniques. In this regard, Rocha et al. [155] propose an automated method for generating conformance diagnostics using declarative constraints derived from an input imperative model. Their method relies on a library of templates internally maintained in the tool. Eligible constraints are generated by verifying the instantiation of those templates against the model's state space. The ones that are behaviorally compatible are then subject to redundancy removal pruning. Finally, the retained constraints are checked for conformance against log traces. In [32], the authors present a tool that derives a set of eligible constraints directly extracting relations based on a selection of BPMN models' activity patterns. The work of Rebmann et al. [153] proposes a framework for extracting best-practice declarative constraints from a collection of imperative models aiming to discover potential violations and undesired behavior. Constraints are extracted akin to [32], then refined and validated via natural-language-processing techniques to measure their relevance for a given event log. Busch et al. [39] adopt a similar technique to check constraints characterizing process model repositories against event logs. All these techniques share our aim to derive declarative constraints from imperative models given as input. However, they do so via simulation or state space exploration, with limited guarantees of behavioral equivalence. In contrast, our work proposes an algorithm that is proven to establish a formal equivalence between the given imperative model and the derived declarative specification. Being based on the sole exploration of the net's structure, it is also lightweight in terms of computational demands.

3.3 Synthesize LTL$_f$ Declarative Specifications from Workflow Nets

In this section, we outline the algorithm (including the three spells to cast: ☆, ☆, and ☆) to synthesize a DECLARE specification $\mathcal{DS}$ from a given input safe and sound Workflow net $\mathcal{WN}$ ensuring behavioral equivalence between them. Algorithm 1 illustrates the transformation process. The algorithm initializes $\mathcal{DS}$ by assigning its alphabet with the transition set of $\mathcal{WN}$ (Alg. 1, ln. 1). Given the Workflow net in Fig. 2.1, e.g., Σ gets $\{t_{\mathsf{a}}, \ldots, t_{\mathsf{g}}, t_{\mathsf{u}}, t_{\mathsf{v}}, t_{\mathsf{w}}\}$. Then, it sets the three (necessary) templates that will be used (ln. 2): AtMostOne (x), End (x), and Alt.Prec. (y, x). A cycle begins to visit all places in $\mathcal{WN}$ and update $\mathcal{DS}$ by including a new constraint per place. Figure 3.1 graphically sketches this passage, which casts the three spells as follows: ☆ If p is the output place as $p\bullet$ is empty, End $(\bullet p)$ is included (ln. 6); ☆ If

Algorithm 1: Wizard's guide to synthesize DECLARE specifications from Workflow nets

Input: $\mathcal{WN} = (P, T, F)$, a safe and sound workflow net;
Output: $\mathcal{DS} = (\text{Rep}, \text{Act}, K)$, a declarative process specification

1 $\Sigma \leftarrow T$; $K \leftarrow \{\}$; # Assign the alphabet with the transition set, and initialize the constraint set
2 $\mathcal{DS} \leftarrow (\{\text{AtMostOne}(x), \text{End}(x), \text{Alt.Prec.}(x, y)\}, \Sigma, K)$ # Initialize $\mathcal{DS}$ including templates
3 **foreach** $p \in P$ **do** # Visit all places in $\mathcal{WN}$
4 **if** $\bullet p \neq \emptyset$ *and* $p\bullet \neq \emptyset$ **then** $K \leftarrow K \cup \{\text{Alt.Prec.}(\bullet p, p\bullet)\}$ # Add the Alt.Prec. $(\bullet p, p\bullet)$ constraint ☆
5 **else if** $\bullet p = \emptyset$ **then** $K \leftarrow K \cup \{\text{AtMostOne}(p\bullet)\}$ # Add the AtMostOne $(p\bullet)$ constraint ☆
6 **else if** $p\bullet = \emptyset$ **then** $K \leftarrow K \cup \{\text{End}(\bullet p)\}$ # Add the End $(\bullet p)$ constraint ☆

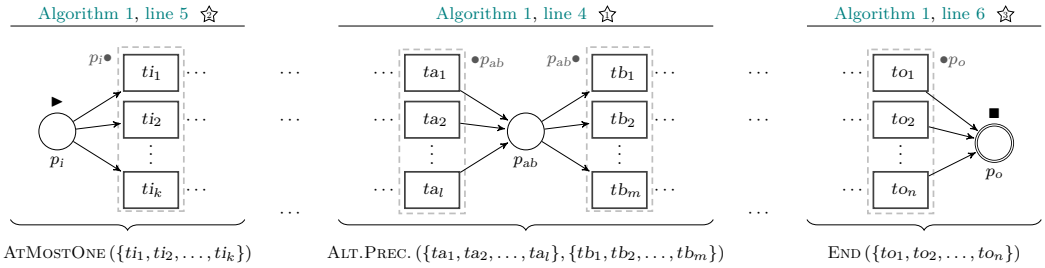


Figure 3.1. A graphical sketch of the execution of Alg. 1

p is the input place as $\bullet p$ is empty, AtMostOne $(p\bullet)$ is added (ln. 5); ☆ Otherwise, Alt.Prec. $(\bullet p, p\bullet)$ becomes one of the constraints in $\mathcal{DS}$ (ln. 4). Intuitively, the rationale is that: ☆ Every time a transition in the postset of p fires, it is necessary that at least one of the transitions in the preset of p fired before and that no transition in the postset of p has fired since then; ☆ Any of the transitions in the postset of ▶ will start the run and will not repeat afterwards (because no firing can assign ▶ with a token again by definition); ☆ Every run must terminate with one of the transitions in the preset of ■.

Table 3.1 shows the constraints that are generated by our algorithm if the Workflow net in Fig. 2.1 is fed as input. It is noteworthy to analyze in particular the non-trivial behavior entailed by constraints that stem from the parsing of places that begin or end cycles like p_8 and p_1 in Fig. 2.1. From the former we derive Alt.Prec. $(t\mathsf{u}, \{t\mathsf{v}, t\mathsf{w}\})$. It states that before $t\mathsf{v}$ or $t\mathsf{w}$, $t\mathsf{u}$ must occur. Also, *neither* $t\mathsf{v}$ nor $t\mathsf{w}$ can recur until $t\mathsf{u}$ is repeated. As a consequence, an *exclusive* choice between $t\mathsf{v}$ and $t\mathsf{w}$ is enforced cyclically for each recurrence of $t\mathsf{u}$. Dually, with Alt.Prec. $(\{t\mathsf{a}, t\mathsf{w}\}, t\mathsf{b})$ (generated by fetching the pre- and post-sets of p_1) we

Table 3.1. DECLARE specification generated from the Workflow net in Fig. 2.1

AtMostOne $(t\mathsf{a})$	End $(t\mathsf{v})$	Alt.Prec. $(\{t\mathsf{a}, t\mathsf{w}\}, t\mathsf{b})$	Alt.Prec. $(t\mathsf{b}, \{t\mathsf{d}, t\mathsf{c}\})$
Alt.Prec. $(\{t\mathsf{d}, t\mathsf{c}\}, t\mathsf{e})$	Alt.Prec. $(t\mathsf{e}, t\mathsf{f})$	Alt.Prec. $(t\mathsf{e}, t\mathsf{g})$	Alt.Prec. $(t\mathsf{f}, t\mathsf{u})$
Alt.Prec. $(t\mathsf{g}, t\mathsf{u})$	Alt.Prec. $(t\mathsf{u}, \{t\mathsf{v}, t\mathsf{w}\})$		

demand that *each* occurrence of $t\mathsf{b}$ follows $t\mathsf{a}$ or $t\mathsf{w}$. From Tab. 3.1 we notice that $t\mathsf{a}$ can occur only once (AtMostOne ($t\mathsf{a}$)), thus the subsequent recurrences of $t\mathsf{b}$ are bound to $t\mathsf{w}$.

Given the construction in Alg. 1, it is clear that the semantics of the resulting Declare specification $\mathcal{DS}$ is an LTL$_f$ formula, traces of which are finite sequences of transitions of the input Workflow net $\mathcal{WN}$. Notice that the mapping of the transitions of $\mathcal{WN}$ to labels (as usual in a process modeling context) can be treated as a post-hoc refinement of $\mathcal{WN}$ and equivalently of $\mathcal{DS}$: Assuming that t_1 maps to label z, e.g., the occurrence of transition t_1 will emit z regardless of the underlying behavioral representation.

As established in the beginning of this section, our goal is to now show the behavioral equivalence between the Declare specification given as output by Alg. 1 and the input safe and sound Workflow net. To this end, we use the following notion of bisimilarity.

Definition 16 (Bisimilarity of Workflow nets and Declare specifications). *A safe and sound Workflow net $\mathcal{WN}$ is* bisimilar *to a Declare specification $\mathcal{DS}$ if and only if the reachability FSA of $\mathcal{WN}$ (as per Def. 8) is bisimilar to the specification FSA of $\mathcal{DS}$ (as per Def. 14).* ◁

Given *(i)* this notion of bisimilarity, and *(ii)* Observation 1, it suffices to show that the two automata accept the same language to prove our claim. This, in turn, means that the Declare specification returned by Alg. 1 accepts all and only the runs of the input safe and sound Workflow net. According to the language-encodability criteria in [95], the obtained bisimilarity ensures operational correspondence between the source and target languages. We now proceed to formally express our claim.

Theorem 1. *Given a safe and sound Workflow net $\mathcal{WN}$, Alg. 1 returns a Declare specification $\mathcal{DS}$ such that: (i) any run of $\mathcal{WN}$ satisfies $\mathcal{DS}$, and (ii) any trace satisfying $\mathcal{DS}$ is a run of $\mathcal{WN}$.* ◁

Proof. We prove that $\mathcal{DS}$ and $\mathcal{WN}$ satisfy the two conditions stated in the claim.

(i) Let σ be a run of $\mathcal{WN}$. We show that $\sigma \models \varphi_{\mathcal{DS}}$. As $\mathcal{WN}$ is a Workflow net, it has a unique input place and a unique output place. Let p_i be ▶ and p_o be ■. In $\varphi_{\mathcal{DS}}$, we have only one constraint for the templates End (x) and AtMostOne (x), namely End $(\bullet p_o)$ and AtMostOne $(p_i\bullet)$. Let $\{to_1, \ldots to_n\}$ be the preset of p_o (with $n \in \mathbb{N}$). Any run of $\mathcal{WN}$ must satisfy $\Box\Diamond(to_1 \vee \ldots \vee to_n)$, i.e., $\varphi_{\text{End}(\bullet p_o)}$, as one of the transitions in the preset of p_o must fire last. Let $\{ti_1, \ldots ti_k\}$ be the postset of p_i (with $k \in \mathbb{N}$). No other place $p' \neq p_i$ can be such that $ti \in p'\bullet$ for any $ti \in p_i\bullet$, otherwise ti would be a dead transition (thus contradicting soundness): The initial marking assigns no token to p', and no marking

except the initial one assigns a token to p_i. As a consequence, any run of $\mathcal{WN}$ must satisfy $\Box((ti_1 \vee \ldots \vee ti_k) \rightarrow \neg \bigcirc \Diamond (ti_1 \vee \ldots \vee ti_k))$, i.e., $\varphi_{\text{AtMostOne}(p_i\bullet)}$.
It remains to show that $\sigma \models \varphi_{\text{Alt.Prec.}(\bullet p, p\bullet)}$ for any arbitrary place $p \in P \setminus \{p_i, p_o\}$. Assume by contradiction that $\sigma \not\models \Box(p\bullet \rightarrow \ominus(\neg p \bullet \mathbf{S} \bullet p))$. Then, there must be a timestep $\ell < |\sigma|$ such that $\sigma, \ell \models p\bullet$, i.e., a transition in $p\bullet$ was fired, but $\sigma, \ell \not\models \ominus(\neg p \bullet \mathbf{S} \bullet p)$. Notice that, as a transition in $p\bullet$ was fired, it means that a transition in $\bullet p$ was fired at a timestep $\ell' < \ell$, as otherwise there would be no token assigned to p at timestep ℓ. Then, for $\sigma, \ell \not\models \ominus(\neg p \bullet \mathbf{S} \bullet p)$ to be true, it must be the case that a transition in $p\bullet$ was fired at some timestep ℓ'' such that $\ell' < \ell'' < \ell$, and no transition in $\bullet p$ has been fired between timesteps ℓ'' and ℓ. However, this, in conjunction with the fact the Workflow net is safe, implies that it would not have been possible to fire a transition in $p\bullet$ at timestep ℓ: p has no token assigned at timestep ℓ as it was consumed to fire a transition in $p\bullet$ at timestep ℓ''. Therefore, $\sigma \models \varphi_{\text{Alt.Prec.}(\bullet p, p\bullet)}$ for any arbitrary place $p \in P \setminus \{p_i, p_o\}$, thus implying that $\sigma \models \varphi_{\mathcal{DS}}$.

(ii) We now show that if a trace σ is such that $\sigma \models \varphi_{\mathcal{DS}}$ then σ is a run of $\mathcal{WN}$. Let p_i be ▶ and p_o be ■ again. Since $\sigma \models \varphi_{\mathcal{DS}}$, we have that the trace correctly ends with a transition in the preset of p_o, because $\sigma \models \varphi_{\text{End}(\bullet p_o)}$. Also, in a run of $\mathcal{WN}$, the transitions in $p_i\bullet$ can only be fired once, otherwise $\bullet p_i$ would be non-empty against the definition of ▶. This holds true in σ, as $\sigma \models \varphi_{\text{AtMostOne}(p_i\bullet)}$. Notice that, unlike all other transitions in $\mathcal{WN}$, only those in $p_i\bullet$ do *not* map to x for Alt.Prec. (y, x) in $\mathcal{DS}$ by design of Alg. 1. Therefore, every trace will begin with the occurrence of one of the transitions in $p_i\bullet$ as it happens with the runs of $\mathcal{WN}$.
It remains to show that every transition in the trace σ was fired in $\mathcal{WN}$ following the preceding sequence of transitions in the trace. Suppose by contradiction that this is not the case, i.e., that there is some transition t fired at a timestep ℓ which could not have been fired in $\mathcal{WN}$ given the prefix of σ from 1 to $\ell - 1$. Then, this implies that at least one of the places p such that $t \in p\bullet$ does not have a token at timestep $\ell - 1$, i.e., $M_{\ell-1}(p) = 0$. Two conditions can entail this situation: Either no transition in $\bullet p$ was fired before, or a transition in $p\bullet$ was fired since the last timestep in which a transition in $\bullet p$ was fired, consuming the only token assigned to p. Both cases contradict the fact that $\sigma \models \varphi_{\text{Alt.Prec.}(\bullet p, p\bullet)}$, thus proving that σ is a valid sequence of transitions with respect to $\mathcal{WN}$. Thus, σ is a run of $\mathcal{WN}$. ⊣

Given Observation 1, we immediately obtain the following corollary.

Corollary 1. *The* Declare *specification $\mathcal{DS}$ given as output by Alg. 1 is bisimilar to the input safe and sound Workflow net $\mathcal{WN}$.* ◁

This result has a profound implication that transcends Declare and LTL$_f$ but pertains to the languages recognized by safe and sound Workflow nets.

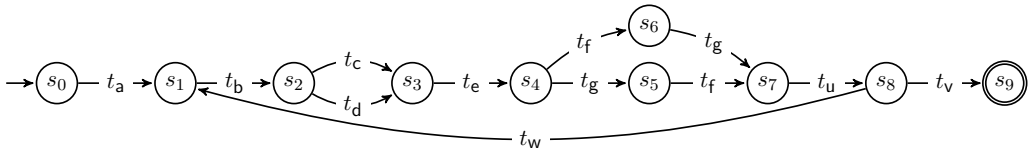


Figure 3.2. FSA of the specification in Tab. 3.1

Theorem 2. *Languages of safe and sound Workflow nets are star-free regular expressions.* ◁

Proof. The claim follows from Theorem 1, recalling that DECLARE patterns are expressed in LTL_f, which is expressively equivalent to star-free regular expressions [58]. ⊣

Space and time complexity. Algorithm 1 outputs a DECLARE specification $\mathcal{DS} = (\mathrm{REP}, \Sigma, K)$ which contains, for each place in the input Workflow net $\mathcal{WN} = (P, T, F)$, a constraint with the pre- and post-sets as its actual parameters. Each transition that is in relation with a place p in the flow relation F appears exactly once in the constraint stemming from p; therefore, the space complexity class of Alg. 1 is $\mathcal{O}(|F|)$. As for the time complexity, we can assume that a pre-processing step is conducted to represent F in the form of a sequence of pairs, associating every place to its pre-set and post-set. The cost of this operation is $\Theta(F)$ and $\mathcal{O}(|P| \times |T|)$. For each place, the algorithm performs up to three if-checks and then a new constraint is created in constant time, hence $\mathcal{O}(|P|)$. Therefore, the time complexity of the algorithm is bounded by the update of K, necessitating up to $\mathcal{O}(|P| \times |T|)$ time.

Next, we experimentally validate and put the above theoretical results to the test.

3.4 Implementation and evaluation

We implemented Alg. 1 in the form of a proof-of-concept prototype encoded in Python. The tool, testbeds, and experimental results are available for public access.[1] In the following, we report on tests conducted with our algorithm's implementation to empirically confirm its soundness, assess memory efficiency, and gauge runtime performance. Finally, we showcase a process diagnostic application as a downstream task for our approach.

3.4.1 Automata bisimulation

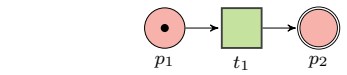
To experimentally validate the correctness of the implementation of Alg. 1, we performed a preliminary comparison of the reachability FSA (Def. 8) of known Workflow nets and the specification FSA (Def. 14) consisting of the DECLARE constraints returned by our tool. Figure 3.2 illustrates the FSA of the specification derived from the Workflow net in Fig. 2.1, computed with a dedicated module presented in [68]. A visual comparison with the reachability FSA of Fig. 2.1 illustrates the bisimilarity guaranteed by Theorem 1. Owing to space constraints, we cannot portray the entire range of automata derived from the Workflow nets in our experiments. The interested reader can find the full collection (including non-free choice nets such as that of [4, Fig. 24]) in our public codebase.[1]

3.4.2 Performance analysis

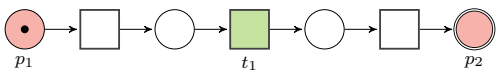
Here, we report on the quantitative assessment of our solution in terms of scalability given an increasing workload, and against real-world testbeds. For the former, we observe the time and space performance of our implemented prototype fed in input with Workflow nets of increasing size. We control the expansion process in two directions, so as to obtain the following separate effects: *(i)* more constraints are generated, while each is exerted on up to three literals; *(ii)* the amount of generated constraints remains fixed, while the literals mapped to their parameters increase. For the real-world testbed, we take as input processes discovered by a well-known imperative process mining algorithm from a collection of openly available event logs. We conducted the performance tests on an AMD Ryzen 9 8945HS CPU at 4.00 GHz with 32 GB RAM running Ubuntu 24.04.1. For the sake of reliability, we ran three iterations for every test configuration and averaged the outputs to derive the final result.

Increasing constraint-set cardinality. To examine the effectiveness of the Alg. 1 in handling an incremental number of constraints, we examine memory utilization and execution time through the progressive rise in the complexity of the input Workflow net. Our evaluation method relies on an expansion mechanism that iteratively applies a structured pattern of four soundness-preserving transformation rules from [3] to progressively increase the number of nodes and their configuration. This leads to a gradual increase in the number of constraints our algorithm needs to initiate. Starting from the Workflow net in Fig. 3.3(a), we designate transition t_1 as a fixed 'pivot', retaining the initial and final places (p_1 and p_2), and iteratively apply the expansion mechanism illustrated in Figs. 3.3(b) to 3.3(e). We apply known workflow patterns in the following order: (Fig. 3.3(b)) We add a transition before

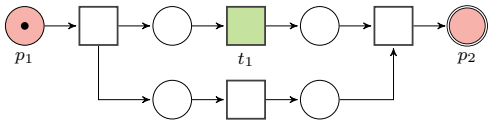
[1] Sp3llsWizard, https://github.com/l2brb/Sp3llsWizard



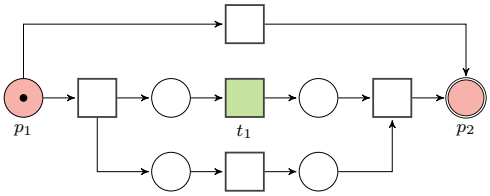
(a) Base net



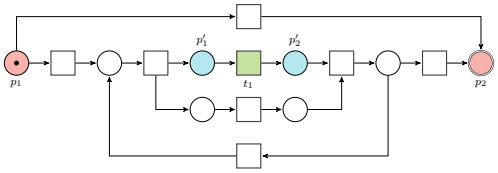
(b) Sequential expansion of the net in Fig. 3.3(a)



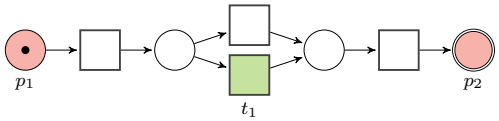
(c) Parallel expansion of the net in Fig. 3.3(b)



(d) Conditional expansion of the net in Fig. 3.3(c)



(e) Loop expansion of the net in Fig. 3.3(d)

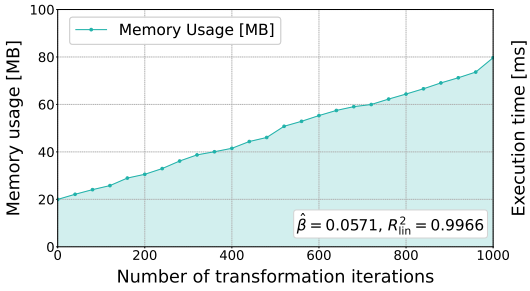


(f) Conditional expansion of the net in Fig. 3.3(b)

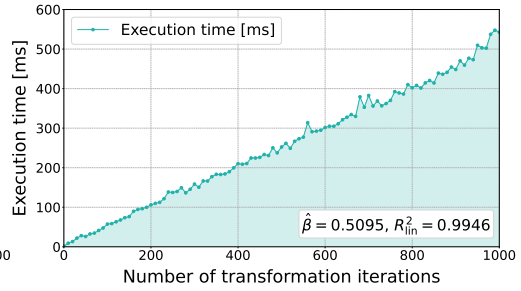
Figure 3.3. Transformation rules used to iteratively expand a safe and sound Workflow net.

and after t_1; (Fig. 3.3(c)) We introduce a parallel execution path; (Fig. 3.3(d)) We insert an exclusive branch; (Fig. 3.3(e)) Finally, we incorporate a loop structure. Upon completion of the expansion process, we execute the algorithm, record the results, and initiate a new iteration, maintaining t_1 unchanged while reassigning p_1 and p_2 with the places that have t_1 in the preset and postset (see the places colored in blue and labeled with p_1' and p_2' in Fig. 3.3(e)). We reiterated the procedure 1000 times.

Figure 3.4 displays the registered memory usage and execution time. To interpret the performance trends, we employ two well-established measures: the coefficient of determination R^2_{lin}, which assesses the goodness-of-fit of the data to a linear

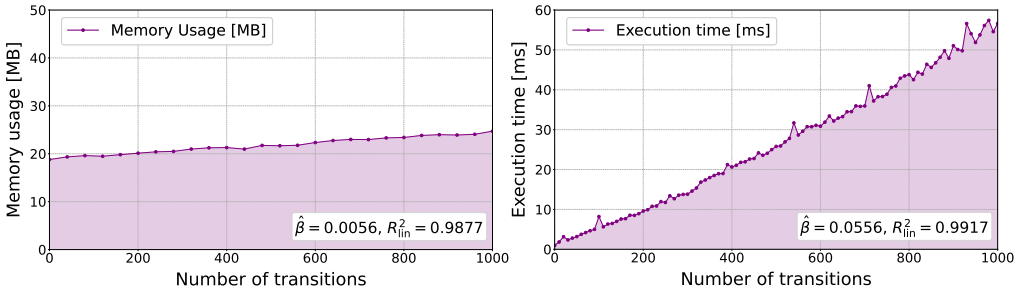


(a) Memory usage



(b) Execution time

Figure 3.4. Test results for the incremental number of constraints setup



(a) Memory usage **(b)** Execution time

Figure 3.5. Test results for the incremental constraints dimension setup

trend, and the $\hat{\beta}$ rate, which serves as a meter for the line's slope. As depicted in Fig. 3.4(a), memory consumption increases linearly with the number of iterations of the expansion mechanism confirmed by $R^2_{\text{lin}} = 0.9966$ and a low slope increase ($\hat{\beta} = 0.0571$). Figure 3.4(b) displays the execution time plot, with $R^2_{\text{lin}} = 0.9946$ and $\hat{\beta} = 0.5095$, thus indicating a linear trend with a slope exhibiting a moderate incline. We remark that these results are in line with the theoretical analysis of the space and time complexity in Sect. 3.3.

Increasing constraint formula size. Here, we configure the test on memory usage and execution time to investigate the algorithm's performance while handling an expanding constraints' formula size (i.e., with an increasing number of disjuncts). To this end, we progressively broaden the Workflow net by applying the soundness-preserving conditional expansion rule from [3] depicted in Fig. 3.3(f) to transition t_1 in the net of Fig. 3.3(b). We reiterate the process 1000 times. Figure 3.5 displays the results we registered. Observing Fig. 3.5(a), we can assert that the memory utilization increases linearly ($R^2_{\text{lin}} = 0.9877$) with a minimal rate ($\hat{\beta} = 0.0056$). The execution time plotted in Fig. 3.5(b) also exhibits a linear increase ($R^2_{\text{lin}} = 0.9917$), with a moderate slope inclination ($\hat{\beta} = 0.0556$). Once more, the results align with the theoretical complexity analysis in Sect. 3.3.

Real-world process model testing. To evaluate the performance of our algorithm in application on real process models, we conduct the same memory usage and execution time tests employing Workflow nets directly derived from a collection of real-life event logs available at *4TU.ResearchData*.[2]

To this end, we employ the Inductive Miner algorithm version proposed in in [117], which filters out infrequent behavior while still discovering well-structured, sound models [22]. Thus, we first run the Inductive Miner on the event logs considered in [22] to generate the Workflow nets. We then apply Alg. 1 to derive the corresponding DECLARE specification. We report the aggregate test result in Tab. 3.2, detailing the

[2]The event logs used in our experiments are publicly available at `https://data.4tu.nl/`

Table 3.2. Performance comparison with real-world process models

Event log	Trans.	Places	Nodes	Mem.usage [MB]	Exec.time [ms]
BPIC 12	78	54	174	19.97	5.11
BPIC 13_{cp}	19	54	44	19.76	1.70
BPIC 13_{inc}	23	17	50	19.89	2.03
BPIC 14_{f}	46	35	102	19.90	3.31
BPIC 15_{1f}	135	89	286	20.44	8.39
BPIC 15_{2f}	200	123	422	20.91	12.30
BPIC 15_{3f}	178	122	396	20.77	11.49
BPIC 15_{4f}	168	115	368	20.55	11.38
BPIC 15_{5f}	150	99	320	20.43	9.16
BPIC 17	87	55	184	19.91	5.67
RTFMP	34	29	82	19.81	3.47
Sepsis	50	39	116	19.75	3.65

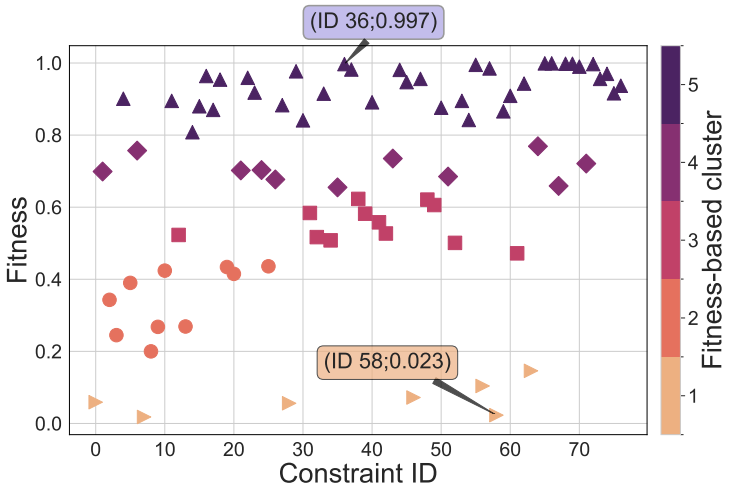


Figure 3.6. Fitness-based clusters of the constraints in the descriptive model of BPIC 15_{5f}

memory usage, the execution time, and all the features of the mined Workflow nets. We find that the overall differences in resource usage are negligible. These real-world test outcomes again follow the complexity assumptions outlined in Sect. 3.3.

3.4.3 Using constraints as determinants for process diagnosis

Algorithm 1 enables the transition from an overarching imperative model to a constraint-based specification, enclosing parts of behavior into separate constraints. Herewith, we aim to demonstrate how we can single out the violated rules constituting the process model behavior, thereby spotlighting points of non-compliance with processes. In other words, we aim to use constraints as determinants for a process diagnosis. For this purpose, we created a dedicated module extending a declarative specification miner for constraint checking via the replay of runs on semi-symbolic

automata like those in Figs. 2.3(a) to 2.3(c), following [69]. Without loss of generality, we build the runs from data pertaining to building permit applications in Dutch municipalities from BPIC 15_{5f} [77] and apply the preprocessing technique mentioned in [22], resulting in 975 traces. We then process the log with the α-algorithm [12] and provide the returned net as input to our implementation of Alg. 1.

We observe that the specification consists of 129 constraints. Of those, our tool detected violations by at least a trace for 77 of those. Figure 3.6 illustrates the percentage of satisfying traces (henceforth, *fitness* for brevity) of the 77 constraints, which we clustered into five distinct groups to ease inspection. We first focus on violated constraints exhibiting high fitness (the blue upward triangles at the top of Fig. 3.6). Let us take, e.g., the constraint identified by ID 36 in the figure: Alt.Prec. $(\{t\texttt{01\_HOOFD\_490\_1}, t\texttt{13\_CRD\_010}\}, t\texttt{01\_HOOFD\_490\_1a})$, which exhibits a fitness of 0.997. This constraint imposes that when "*Set Decision Status*" (t01_HOOFD_490_1a) occurs, it should be preceded by either "*Create Environmental Permit Decision*" (t01_HOOFD_490_1) or "*Coordination of Application*" (t13_CRD_010). In the three traces violating the constraint (11369696, 9613229, 12135936), though, "*Set Decision Status*" is preceded by neither of the two. By further inspection, we observe "*No Permit Needed or Only Notification Needed*" (t14_VRIJ_010) in the trace prefix instead, suggesting that the process bypasses the standard decision-making steps defined by the reference model in favor of an alternative where a permit decision is unnecessary. On the other side of the spectrum, let us look at constraints with low trace fitness values (depicted by rightward orange triangles in Fig. 3.6). These constraints likely suffer from systematic defects rather than spurious alterations in the process behavior. Constraint ID 58, e.g., belongs to this group: Alt.Prec. $(t\texttt{1\_HOOFD\_510\_2}, \{t\texttt{01\_HOOFD\_510\_3}, t\texttt{01\_HOOFD\_520}, t\texttt{END}\})$ (depicted in the lower section of Fig. 3.6). Other constraints in the same group have in common the presence of tEND in the activator's set. An explanation is that the BPIC 15 log allows a multitude of possible conclusions. The α-algorithm, though, disregards the occurrence frequency of individual transitions during model construction, resulting in a non-selective inclusion of all events. Consequently, this affects the fitness of those constraints. Our tool specifically pinpoints and isolates the effect of this tendency from the remainder of the net.

Chapter 4

Measuring LTL$_f$ Process Specifications

4.1 Introduction

The declarative specification of a process allows users and designers to norm and control its behavior through rules. These rules consist of temporal logic formulae (such as LTL$_f$, see Sect. 2.1.2) that are verified against recorded runs of the process-aware systems in an event log, to check their compliance with the behavioral properties it must guarantee. This automated checking task shows wide adoption in multiple areas of computer science, including process mining [144, 92], planning [23, 41], and software engineering [119, 42]. Rule-based specifications allow for the definition of system and process behavior focusing on core constraints that must be satisfied, leaving other execution details to knowledge- and context-based decisions of the actors [71] with a so-called "open-ended" paradigm [130]. This aspect makes them particularly suitable for scenarios in which workflow flexibility is key [70, 11], as in the healthcare domain [156, 140] with the specification of clinical practice guidelines [97, 133].

Despite the increasing interest in this challenge, we observe that a fundamental problem remains unaddressed. Measuring the extent to which traces adhere to the admissible behavior in terms of *specifications*, or sets of rules, is still a problem that leaves ample margins for investigation.

> **RQ2: Conformance Measurement**
>
> How can the satisfaction of declarative process specifications be quantitatively assessed over observed executions?

Let us consider an example inspired by [104] on the clinical pathway for the

treatment of unstable angina. A declarative process specification $\mathcal{S}$, e.g., consists of two rules, Ψ_1 and Ψ_2. Rule Ψ_1 states that "Whenever a specific combination of medications is prescribed, it should be preceded by an electrocardiogram (ECG) and a cardiac enzyme test." Rule Ψ_2 indicates that "If a patient experiences severe chest pain, an immediate administration of sublingual nitroglycerin will follow." Confidence is the ratio of events in a log that satisfy the consequent (i.e., the *target*, like the ECG and cardiac enzyme test in Ψ_1), given the satisfaction of the antecedent (i.e., the *activator*, like the occurrence of severe chest pain in Ψ_2).

Suppose the confidence of Ψ_1 and Ψ_2 are 100 % and 50 %, respectively. What is the confidence of $\mathcal{S} = \{\Psi_1, \Psi_2\}$? Considering the confidence of a single formula consisting of the conjunction of all rules (e.g., $\Psi_1 \wedge \Psi_2$) may be too coarse-grained, since violating a single rule in one event or in multiple ones would lead to the same result of violating the whole specification for that trace. Similarly, aggregating measures over multiple rules (as in [44]) may be misleading. Let the activator of Ψ_1, e.g., occur 100 times in the log, always leading to the satisfaction of the rule (Confidence 100 %), and the activator of Ψ_2 occur twice, leading to the violation of the rule once (Confidence 50 %). Consequently, there will be one violation out of 102 occurrences of the activators. Yet, the average confidence of the two rules will be 75 %. A lack of a proper framework to gauge the degree to which specifications are satisfied by, or emerging from, recorded data, hinders their adoption for the discovery, checking, and drift-detection of system and process behavior in rule-based settings. In turn, this issue slows down the evolution of dedicated techniques even where those tasks may turn out to be pivotal, like in the highly dynamic and flexible context of clinical pathways.

To overcome this issue, we propose a new approach adapting and extending the concept of Reactive Constraint (RCon), originally proposed in [45], and the measurement framework for single declarative rules expressed as RCons [44]. RCons are rules expressed in an if–then fashion (like Ψ_1 and Ψ_2), namely a pair of LTL$_f$ formulae, one of which is the activator (if) and the other is the target (then). RCons cover the full spectrum of declarative process specification languages, such as DECLARE [144], as any LTL$_f$ formula can be translated into an RCon [45].

Equipped with this notion, we propose a measurement framework that takes inspiration from classical association rule mining [91] to assess whether, and in how far, process specifications consisting of LTL$_f$-based rules expressed in an "if–then" fashion are satisfied by a trace. Our approach is rooted in probability theory and statistical inference. Specifically, in order to provide a non-binary interpretation for specification measurements, we model events of satisfaction and violation of formulae by traces (and logs of traces) using probability theory, and

derive corresponding maximum-likelihood estimators for these probabilistic models. Moreover, we show that these estimators can be computed in polynomial time. To the best of our knowledge, this work is the first to tackle and solve the problem of devising well-defined measures for *entire* declarative specifications consisting of multiple rules. To tackle this problem, we move from an ad-hoc counting approach to a sound probabilistic theory based on maximum likelihood estimation. Finally, we conduct an evaluation of our approach on real-world data with its software prototype implementation.

4.2 Related Work

Different contributions in the literature aim at quantitative extensions of LTL/LTL$_f$, enriching the languages with quantitative operators. The work by [16], [115], and [148] proposed the addition of quantitative operators into the logic. LTL[$\mathcal{F}$] [16] introduces quality operators quantifying over distinct satisfactions of a formula. Quantified-LTL [148] uses quantifiers over its propositional variables, also in probabilistic systems such as Markov chains.

In [115], the quantification of satisfaction, applied in the context of planning, is based on associating costs to specification violations based on user ranking of tasks priority. Differently from these methods, we do not extend the syntax and semantics of LTL$_f$ with new operators, as we quantify the satisfaction of formulae based on standard LTL$_f$.

As for the interplay of temporal logic and probabilities, statistical model checking techniques [118] retrieve the probability for a formula to be satisfied in a probabilistic environment as Markov chains. Their goal is to predict the likelihood of a formula for any possible execution (a probabilistic relaxation of traditional model checking), while we study only already executed traces. The method proposed in [132] is close to our investigation, as it resorts to the association of a probability threshold to each rule. The threshold is used to perform relaxed conformance checking: each rule should hold in at least a portion of the log that is greater than that value. However, only single rules are analyzed and the trace satisfaction is not quantified but considered as boolean, whereas we can assess the partial satisfaction of specifications, also on single traces.

A dual perspective on quantitative analysis of declarative specifications is taken in [50], where the authors address the problem of measuring inconsistency of LTL$_f$-based process specifications. They introduce a paraconsistent semantics for LTL$_f$ over fixed traces and define element-based inconsistency measures atop it, later extending the formalism with preference relations among formulae. While their goal

is to quantify how internally contradictory a specification is, independently of any log, our framework targets the complementary problem of quantifying the degree to which an event log satisfies a (consistent) specification, while retaining the standard LTL$_f$ semantics.

In process mining, the compliance of process models to the data is usually gauged with four scores: Fitness, Precision, Generalization, and Simplicity [36]. In [59] Fitness, Precision, and Generalization are devised for DECLARE models through alignments, while in [151] Fitness and Precision are computed for any regular language through entropy. Our framework focuses on a different set of measures, inspired by association rule mining. The comparison and integration of the four measures above paves the path for future research endeavors. Notably, the novel measure of informativeness is proposed in [38] to understand the differences between compliant traces. We showed in Sect. 4.9 how separate measures can spot differences in compliant specifications, thus a deeper analysis in this direction is an interesting research outlook.

4.3 Event logs and Linear Temporal Logic on Finite Traces LTL$_f$

As introduced in Sect. 2.2, we define collections of traces representing multiple executions of the same process as an *event log*. Formally, an event log is a multiset of traces where each trace represents one run so that recurring executions are captured by their multiplicities, as per Def. 15.
Table 4.1 presents a log $L = \{\pi_1^{17}, \pi_2^{6}, \pi_3^{5}, \pi_4^{12}, \pi_5^{5}\}$ defined over alphabet $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}\}$. Its cardinality is 45.

Theorem 3 ([90]). *Let t be a finite trace of length $n \in \mathbb{N}$. Checking whether (π, i) (with $1 \leq i \leq n$) satisfies an LTL$_f$ formula φ, $(\pi, i) \models \varphi$, is feasible in $O(n^2 \times \|\varphi\|)$.* ◁

Proof. It follows from the proof in [90] elaborated for future operators ($\bigcirc$, $\Diamond$, $\Box$, and $\mathbf{U}$). Notice that the use of past modalities $\ominus$, $\boxminus$, $\diamondminus$ and $\mathbf{S}$ do not alter the complexity. Indeed, they can be included in the parse tree of the constructive proof in [90] as the respective future counterparts and checked against the trace read in reverse (i.e., from end to start [45]). ⊣

Corollary 2. *Let L be an event log as per Def. 15 consisting of $m \in \mathbb{N}$ distinct traces of length up to $n \in \mathbb{N}$ and cardinality $|L| \geq m$. Labeling the events in L that satisfy an LTL$_f$ formula φ is feasible in $O(n^3 \times \|\varphi\| \times m)$.* ◁

Proof. The proof follows from Theorem 3: the checking is done for the $O(n)$ events of all m distinct traces in L. ⊣

Table 4.1. Measurements of RCons $\Psi_1 = \mathtt{c} \Box\!\!\rightarrow \Diamond \mathtt{a}$, and $\Psi_2 = \mathtt{d} \Box\!\!\rightarrow \Diamond \mathtt{e}$, and of specification $\mathcal{S} = \{\Psi_1, \Psi_2\}$ on a log.

Log	Evaluation	RCon / Specification	P of RCon / P of $\mathcal{S}$	P of act.	P of target	Support	Confidence	Recall	Specificity	Lift
$\pi_1 = \langle \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{b}, \mathtt{c}, \mathtt{e}, \mathtt{c}, \mathtt{b} \rangle$	$\langle \times, \times, 1, \times, \times, 1, \times, 1, \times \rangle$	$\Psi_1 = \mathtt{c} \Box\!\!\rightarrow \Diamond \mathtt{a}$	1.00	0.33	1.00	0.33	1.00	0.33	0.00	1.00
	$\langle \times, \times, \times, 1, \times, \times, \times, \times, \times \rangle$	$\Psi_2 = \mathtt{d} \Box\!\!\rightarrow \Diamond \mathtt{e}$	1.00	0.11	0.78	0.11	1.00	0.14	0.25	1.29
	$\langle \times, \times, 1, 1, \times, 1, \times, 1, \times \rangle$	$\mathcal{S} = \{\Psi_1, \Psi_2\}$	1.00	0.44	0.89	0.44	1.00	0.50	0.20	1.13
$\pi_2 = \langle \mathtt{b}, \mathtt{d}, \mathtt{a}, \mathtt{b}, \mathtt{b}, \mathtt{d}, \mathtt{e}, \mathtt{d}, \mathtt{c} \rangle$	$\langle \times, \times, \times, \times, \times, \times, \times, \times, 1 \rangle$	$\Psi_1 = \mathtt{c} \Box\!\!\rightarrow \Diamond \mathtt{a}$	1.00	0.11	0.78	0.11	1.00	0.14	0.25	1.29
	$\langle \times, 1, \times, \times, \times, 1, \times, 0, \times \rangle$	$\Psi_2 = \mathtt{d} \Box\!\!\rightarrow \Diamond \mathtt{e}$	0.67	0.33	0.78	0.22	0.67	0.29	0.17	0.86
	$\langle \times, 1, \times, \times, \times, 1, \times, 0, 1 \rangle$	$\mathcal{S} = \{\Psi_1, \Psi_2\}$	0.75	0.44	0.78	0.33	0.75	0.43	0.20	0.96
	$\langle 0, \times, \times, \times, 1, \times, \times, 1, \times, 1 \rangle$	$\Psi_1 = \mathtt{c} \Box\!\!\rightarrow \Diamond \mathtt{a}$	0.75	0.40	0.80	0.30	0.75	0.38	0.17	0.94
$\pi_3 = \langle \mathtt{c}, \mathtt{d}, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{e}, \mathtt{b}, \mathtt{c}, \mathtt{b}, \mathtt{c} \rangle$	$\langle \times, 1, \times, \times, \times, \times, \times, \times, \times, \times \rangle$	$\Psi_2 = \mathtt{d} \Box\!\!\rightarrow \Diamond \mathtt{e}$	1.00	0.10	0.60	0.10	1.00	0.17	0.44	1.67
	$\langle 0, 1, \times, \times, 1, \times, \times, 1, \times, 1 \rangle$	$\mathcal{S} = \{\Psi_1, \Psi_2\}$	0.80	0.50	0.70	0.40	0.80	0.57	0.40	1.14
	$\langle \times, 0, \times, 1, \times, \times \rangle$	$\Psi_1 = \mathtt{c} \Box\!\!\rightarrow \Diamond \mathtt{a}$	0.50	0.33	0.67	0.17	0.50	0.25	0.25	0.75
	$\langle \times, \times, \times, \times, \times, \times \rangle$	$\Psi_2 = \mathtt{d} \Box\!\!\rightarrow \Diamond \mathtt{e}$	NaN	0.00	0.83	0.00	NaN	0.00	0.17	NaN
$\pi_4 = \langle \mathtt{b}, \mathtt{c}, \mathtt{a}, \mathtt{c}, \mathtt{e}, \mathtt{a} \rangle$	$\langle \times, 0, \times, 1, \times, \times \rangle$	$\mathcal{S} = \{\Psi_1, \Psi_2\}$	0.50	0.33	0.50	0.17	0.50	0.33	0.50	1.00
	$\langle \times, \times, \times \rangle$	$\Psi_1 = \mathtt{c} \Box\!\!\rightarrow \Diamond \mathtt{a}$	NaN	0.00	0.00	0.00	NaN	NaN	1.00	NaN
$\pi_5 = \langle \mathtt{b}, \mathtt{b}, \mathtt{b} \rangle$	$\langle \times, \times, \times \rangle$	$\Psi_2 = \mathtt{d} \Box\!\!\rightarrow \Diamond \mathtt{e}$	NaN	0.00	0.00	0.00	NaN	NaN	1.00	NaN
	$\langle \times, \times, \times \rangle$	$\mathcal{S} = \{\Psi_1, \Psi_2\}$	NaN	0.00	0.00	0.00	NaN	NaN	1.00	NaN
$L = \{\pi_1^{17}, \pi_2^6, \pi_3^5, \pi_4^{12}, \pi_5^5\}$ $\lvert L \rvert = 45$		$\Psi_1 = \mathtt{c} \Box\!\!\rightarrow \Diamond \mathtt{a}$	0.80	0.27	0.75	0.22	0.80	0.29	0.27	1.07
		$\Psi_2 = \mathtt{d} \Box\!\!\rightarrow \Diamond \mathtt{e}$	0.85	0.10	0.69	0.08	0.85	0.12	0.33	1.24
		$\mathcal{S} = \{\Psi_1, \Psi_2\}$	0.81	0.37	0.65	0.30	0.81	0.46	0.44	1.25

NaN values denote a division by 0.

4.4 Reactive Constraints (RCons) and process specifications

In this section, we illustrate how we adopt LTL$_f$ to express rules specifying the behavior of a process in the form of Reactive Constraints (RCons).

4.4.1 Reactive Constraints (RCons)

RCons express LTL$_f$-based rules as antecedent-consequent pairs in an "if-then" fashion. Next, we formalize their definition.

Definition 17 (Reactive Constraint (RCon)). *Given an alphabet of propositional symbols Σ, let φ_α and φ_τ be LTL$_f$ formulae over Σ. A* Reactive Constraint (RCon) *Ψ is a pair $(\varphi_\alpha, \varphi_\tau)$ hereafter denoted as $\Psi \triangleq \varphi_\alpha \boxdot\!\!\rightarrow \varphi_\tau$. We name φ_α as* activator *and φ_τ as* target. ◁

We define the semantics of an RCon $\Psi = \varphi_\alpha \boxdot\!\!\rightarrow \varphi_\tau$ as follows: given a trace π of length n and an instant i with $1 \leq i \leq n$, we say that
Ψ is satisfied by π in i, i.e., $\pi, i \models \Psi$, iff $\pi, i \models \varphi_\alpha$ and $\pi, i \models \varphi_\tau$
Ψ is violated by π in i, $\pi, i \not\models \Psi$, iff $\pi, i \models \varphi_\alpha$ and $\pi, i \not\models \varphi_\tau$
Ψ is unaffected by π in i, iff $\pi, i \not\models \varphi_\alpha$.
We also say that Ψ **is activated by** π if there exists an instant i s.t. $1 \leq i \leq n$ and $\pi, i \models \varphi_\alpha$. For example, take $\Psi_1 = \mathsf{c} \boxdot\!\!\rightarrow \Diamond \mathsf{a}$ in Tab. 4.1. At every occurrence of c (the activator), Ψ_1 is either *satisfied* (if c is eventually preceded by a as $\Diamond \mathsf{a}$ is the target), or *violated*. Ψ_1 is instead *unaffected* by those events in which c does not occur. The activator of $\Psi_2 = \mathsf{d} \boxdot\!\!\rightarrow \Diamond \mathsf{e}$ in Tab. 4.1 is d and the target is $\Diamond \mathsf{e}$. Whenever d occurs, it is either satisfied (if eventually followed by e) or violated (otherwise). It is unaffected by events wherein d does not hold. Notice that by declaring that the activator of Ψ_1 is c, the user makes the "trigger" of the rule explicit. Ψ_1 and Ψ_2 are the RCon representation of what are known as Precedence (c, a) and Response (d, e) in the declarative process specification language Declare [144], respectively.

A Note on the Role of the Activators in RCons. We remark that any LTL$_f$ formula ϕ can be expressed by means of an RCon. The expressiveness of RCons fully covers that of LTL$_f$ and, a fortiori, of Declare. The examination of the expressiveness of RCons is discussed in detail in [45, 44]. On the other hand, the temporal conditions that an LTL$_f$formula such as $\Box(\phi_1 \rightarrow \phi_2)$ exerts do not substantially differ from those of $\phi_1 \boxdot\!\!\rightarrow \phi_2$. However, we adopt RCons to express rules in a mining context because the explicit definition of the activator makes it possible to distinguish an interesting satisfaction from a vacuous one [114]. The

LTL_f formulae $\psi_1 = \Box(\mathsf{c} \rightarrow \Diamond\mathsf{a})$ and $\psi_2 = \Box((\neg\mathsf{c}\ \mathbf{U}\ \mathsf{a}) \vee \neg\mathsf{c})$ both express the Precedence (c, a) constraint. The rule dictates that, for every event, *if* c occurs, then it must be preceded by a. This explanation closely resembles ψ_1. Therefore, if c does not occur, the constraint is also satisfied (regardless of whether a occurs or not: *ex falso sequitur quodlibet*), though vacuously. Notice that the latter condition is explicit in the second conjunct under $\Box$ in ψ_2: "$\ldots \vee \neg\mathsf{c}$". Indicating that a rule is satisfied though being unable to distinguish whether it is actually triggered or not adds limited information [131]. The adoption of RCons, though not solving the vacuity problem (activator and target could still be vacuously satisfiable formulae per se), lets *the user* explicitly define the conditions that make a satisfaction interesting. With $\mathsf{c} \,\square\!\!\rightarrow \Diamond\mathsf{a}$, e.g., we state that the rule is unaffected (neither violated nor satisfied) by events in which c does not occur. Notice, however, that alternative expressions can be used to customize the interpretation of the rules. To adopt the classical binary interpretation of LTL_f formulae, Precedence (c, a) can be expressed, e.g., as $\Psi_3 = \mathit{True} \,\square\!\!\rightarrow ((\neg\mathsf{c}\ \mathbf{U}\ \mathsf{a}) \vee \neg\mathsf{c})$, or $\Psi_4 = \pi_{\text{Start}} \,\square\!\!\rightarrow \Box(\mathsf{c} \rightarrow \Diamond\mathsf{a})$. In the first case, Ψ_3 indicates that every event activates the RCon (because *True* holds in every event). Therefore, the satisfaction of the target determines the satisfaction of the constraint. In the second case, the activator of Ψ_4 is π_{Start}. Thus, the first event acts as a representative for the whole trace as either satisfying (if no events in it violates the constraint) or violating (if at least one event violates it). The constraint is unaffected by the trace in every other event. The definition of well-formed RCons and guidelines for their formulation in a mining context go beyond the scope of this paper, and suggest interesting theoretical investigation for future work.

4.4.2 Process specifications

Equipped with the above notion of RCons and the rationale behind their use in this context, we can define a process specification as follows.

Definition 18 (Rule-based LTL_f process specification). *A rule-based LTL_f process specification (henceforth,* specification *for short) is a finite non-empty set of RCons $\mathcal{S} \triangleq \{\Psi_1, \ldots, \Psi_s\}$, with $s \in \mathbb{N}$.* ◁

Table 4.1 presents a specification $\mathcal{S} = \{\Psi_1, \Psi_2\}$ composed by the Ψ_1 RCon above and $\Psi_2 = \mathsf{d} \,\square\!\!\rightarrow \Diamond\mathsf{e}$.

Corollary 3. *Let $\mathcal{S} = \{\Psi_1, \ldots, \Psi_s\}$ be a specification consisting of s RCons, the activator and target of which are of size up to $\|\varphi\|$. Labeling the events in L with the satisfaction of activator and target of every RCon in $\mathcal{S}$ is feasible in $O(n^3 \times \|\varphi\| \times |L| \times s)$.* ◁

Proof. The proof follows from Corollary 2, as a pair of labels is sufficient for all RCons in the specification. ⊣

Take, e.g., trace $\pi_4 = \langle \mathsf{b}, \mathsf{c}, \mathsf{a}, \mathsf{c}, \mathsf{e}, \mathsf{a} \rangle$ from Tab. 4.1 and the aforementioned RCon $\Psi_1 = \mathsf{c} \boxdot\!\!\rightarrow \Diamond \mathsf{a}$. The activator ($\mathsf{c}$) is satisfied in $(\pi_4, 2)$ and $(\pi_4, 4)$. We can thus label every event in π_4 thereby creating a new sequence as follows: $\langle 0, 1, 0, 1, 0, 0 \rangle$ where 1 and 0 indicate a satisfaction and a violation of the formula in the corresponding event, respectively. Similarly, we can create a sequence of labels denoting whether the target ($\Diamond \mathsf{a}$) is satisfied: $\langle 0, 0, 1, 1, 1, 1 \rangle$.

A trivial approach to classify traces as compliant with an RCons or not is to check whether no event violates it. Nevertheless, especially in checking contexts, understanding the *extent* to which a trace and a log satisfy a specification is key [57]. Next, we introduce the interestingness measures, i.e., the quantitative device we employ to quantify the extent of specification satisfaction. Subsequently, we lay the foundations rooted in probabilistic theory to reach this goal.

4.5 Statistical preliminaries

4.5.1 Basic Probability Notation

The theory of probability is a fundamental mathematical concept that underpins a wide range of fields, including statistics, economics, and physics [89]. The key concepts in probability theory include the sample space Ω, events $A, B, C, \ldots$, probability functions $P(\cdot)$, and conditional probabilities $P(A \mid B)$. The sample space Ω represents the set of all possible outcomes of an experiment. For example, when flipping a coin, the sample space is $\{\mathrm{H}, \mathrm{T}\}$, where H represents heads and T represents tails.

Events $A, B, C, \ldots$ in probability theory are subsets of the sample space ($A \cup B \cup C \ldots \subseteq \Omega$) representing specific outcomes of interest. For example, A could represent the event of getting heads when flipping a coin. Probability functions $P(\cdot)$ assign probabilities to events: $P(A)$, e.g., represents the probability of event A occurring, and it is a number between 0 and 1, inclusive. $P(A \cap B)$ represents the probability of both A and B occurring, also known as the intersection of A and B. $P(A \cup B)$ represents the probability of either A or B occurring, also known as the union of A and B. Notice that the notion of event in probability slightly differs from that of event in the context of process and specification mining (see Def. 15). $P(A)$ represents the probability that a record in the trace satisfies A. In this setting, then, A is a statement that can hold true, or not, in the elements that a trace consists of.

4.5.2 Conditional Probability

Conditional probability is the probability of an event A given that another event B has occurred. We write it as $P(A \mid B)$ and define it using Kolmogorov's analysis:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}. \tag{4.1}$$

This formula allows us to update our beliefs about the occurrence of an event based on new information. The following properties of conditional probability hold. Firstly, $P(A \mid B) \geq 0$ for all events A and B. This means that the conditional probability of A given B is always non-negative.

Secondly, $P(\Omega \mid B) = 1$ for any event B. This means that the probability of the entire sample space given that B has occurred is equal to 1. Thirdly, if A and B are mutually exclusive events (i.e., $A \cap B = \emptyset$), then $P(A \mid B) = 0$. This is because if A and B cannot occur simultaneously, then the occurrence of B rules out the possibility of A occurring.

The law of total probability (LTP) can be used to compute conditional probabilities. If $A_1, A_2, \ldots, A_n$ are mutually exclusive events that partition the sample space, then

$$P(B) = \sum_{i=1}^{n} P(B \mid A_i)P(A_i). \tag{4.2}$$

This formula can be rearranged to compute conditional probabilities as follows (for any $1 \leq j \leq n$):

$$P(A_j \mid B) = \frac{P(B \mid A_j)P(A_j)}{\sum_{i=1}^{n} P(B \mid A_i)P(A_i)}. \tag{4.3}$$

4.5.3 Discrete Random Variables

A discrete random variable is a random variable that takes on a countable number of values, such as the number of heads obtained in a series of coin flips, or the number of defects in a batch of products. We denote a discrete random variable as X and its possible values as $x_1, x_2, \ldots, x_n$. The probability distribution of X specifies the probabilities $P(X = x_k)$ for each possible value x_k with $1 \leq k \leq n$. For random variables, the conditional probability is defined as

$$P(X = x|Y = y) = \frac{P(X = x \cap Y = y)}{P(Y = y)} \tag{4.4}$$

where X and Y are discrete random variables, and x and y are possible values that X and Y can take, respectively. $P(X = x \cap Y = y)$ is the probability that $X = x$ and $Y = y$ occur simultaneously, and $P(Y = y)$ is the probability that $Y = y$ occurs.

The law of total probability for discrete random variables can be written as follows. Let X be a discrete random variable and let $Y_1, Y_2, \ldots, Y_m$ be a partition of the sample space Ω, i.e., $\Omega = Y_1 \cup Y_2 \cup \cdots \cup Y_m$ and $Y_i \cap Y_j = \emptyset$ for $1 \leq i \leq m$, $1 \leq j \leq m$, and $i \neq j$. Then, for any event A, the law of total probability states:

$$P(A) = \sum_{i=1}^{m} P(Y_i)P(A|Y_i) \tag{4.5}$$

where the sum is taken over all possible values of i.

4.5.4 Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) is a popular statistical method for estimating the parameters of a statistical model. MLE is widely used in many fields, including econometrics, biostatistics, and engineering, due to its desirable properties [43]. One of the key strengths of MLE is its asymptotic efficiency, which means that as the sample size increases, the MLE estimates converge to the true parameter values at the fastest possible rate among all consistent estimators. This means that MLE produces the most precise estimates possible, given the available data.

Furthermore, MLE is a consistent estimator, meaning that as the sample size increases, the MLE estimates converge to the true parameter values. As a result, the accuracy of the MLE estimates increases with more data becoming available. Under certain conditions, MLE is also an unbiased estimator, meaning that the expected value of the estimates is equal to the true parameter value. This requires that the model is correctly specified and the sample size is sufficiently large.

Finally, MLE has solid theoretical foundations based on sound statistical theory and has been extensively studied in the literature, providing a large body of knowledge for understanding its properties and effective use. Additionally, the computation of MLE is widely supported by several existing software packages encoded in R and Python, among others.

4.6 Interestingness measures for Reactive Constraint (RCon)

The association rule mining field has a long history of measures development for association rules, also called interestingness measures [14, 91]. These rules have a standard "if-A-then-B" form, where A and B are elements that may occur in a phenomenon, e.g., instructions in a set of database transactions, or events in a set of process traces. These measures are based on the (joint) probabilities of the occurrences (and co-occurrences) of A and B with the general goal of understanding

Table 4.2. A selection of interestingness measures

Measure	Definition
Support	$P(A \cap B)$
Confidence	$P(B\|A)$
Recall	$P(A\|B)$
Specificity	$P(\neg B\|\neg A)$
Lift	$\frac{P(A \cap B)}{P(A)P(B)}$

whether there is a significant (directional) relation between the two elements or, alternatively, whether their co-occurrence is uncorrelated. Historically, the development of these measures began in market basket analysis with the introduction of the A-Priori algorithm [15]. The problem that the algorithm addressed was to discover, given a set of transactions, association rules between sets of frequently co-occurring elements. Yet, the simple co-occurrence does not necessarily imply a correlation between the elements. Therefore, more refined measures that distinguish authentic associations from spurious ones were developed [91].

Let us consider a few examples of such measures, presented in Tab. 4.2: Given an "if-A-then-B" rule, the Support measure, $P(A \cap B)$, quantifies the joint occurrences of the two elements; Confidence, $P(B|A)$, considers the occurrence of B only when we know that A occurred; Specificity, $P(\neg B|\neg A)$, measures the non-occurrence of B given the non-occurrence of A; Recall, $P(A|B)$, measures the conditional occurrence of A given B; finally, Lift, $\frac{P(A \cap B)}{P(A)P(B)}$, is the ratio of the probability of co-occurrence of A and B over the product of the individual probabilities of A and B to occur. Intuitively, Support is high when A and B occur frequently together; Confidence is high when B occurs every time A occurs, while Recall is high in the opposite situation, i.e., if A occurs every time B occurs; Specificity is high when B does not occur if A does not occur; Lift is high when the separate probability of A and B occurring is lower than the probability of their joint occurrence.

Clearly, these measures quantify different aspects of the rule and, depending on the needs of the user, one measure may turn out to be more useful than another. A plethora of other interestingness measures have been defined in the literature. We refer the reader to existing surveys for an extensive overview [91, 121, 168].

In the context of process mining, it has already been shown how to apply these interestingness measures to single RCon rules [44]. By definition, RCons are "if-then" rules too, which makes these measures suitable for interestingness measurement based on the probabilities of the satisfaction of the activator and target conditions in a trace or a log.

The main goal of this chapter is to extend these results to specifications of RCons.

To this end, we must assume a probabilistic model and derive sound statistical estimators for specifications satisfied by traces and, subsequently, logs.

4.7 Estimators for LTL$_f$ formulae

The interestingness measures for RCons are based on the probabilities of their activator (φ_α) and target (φ_τ) LTL$_f$ formulae. In this section, we propose estimators for the probabilities of traces and logs satisfying LTL$_f$ formulae and show that these estimators are computable in polynomial time.

4.7.1 Trace estimators

We start by defining probabilistic models for the evaluation of formulae over traces. One can consider the probability of an event in a given trace $\pi = \langle e_1, \ldots, e_n \rangle$ to satisfy an LTL$_f$ formula φ as the degree to which φ is satisfied in that trace, which we denote as $P(\varphi(\pi))$.

Throughout this section, we assume the existence of a labeling mechanism Λ that, when given an event e in a trace π and a formula φ, marks the event with 1 if the event satisfies φ or with 0 otherwise, i.e., $\Lambda(e, \varphi) \in \{0, 1\}$. This procedure can be achieved in polynomial time as shown in Corollary 2 through automata-based techniques for LTL$_f$ formulae verification [44].

Therefore, every trace π can be associated with a binary sequence $x_{\varphi,t}$ as follows:

$$x_{\varphi,t} = \langle \Lambda(e_1, \varphi), \ldots, \Lambda(e_n, \varphi) \rangle .$$

In what follows, we assume that $P(\varphi(\pi))$ (the probability of π to satisfy φ), is independent of the position of the event in the trace. This is an uninformative prior assumption, i.e., we assume that we are unaware of the values of other events and of the event location within the trace when evaluating a specific event. Thus, the sequence $x_{\varphi,t}$ can be viewed as an independent and identically distributed (i.id.) draw from a Bernoulli random variable $X_{\varphi,t}$, which takes the value of 1 with probability $P(\varphi(\pi))$ and 0 otherwise, i.e.,

$$X_{\varphi,t} = \begin{cases} 1, & \text{w.p. } P(\varphi(\pi)), \\ 0, & \text{otherwise.} \end{cases} \tag{4.6}$$

This leads us to our first estimator, namely that of $P(\varphi(\pi))$.[1]

[1] We write $P(\varphi(\pi))$ for the probability of specific events such as the satisfaction of a formula, and $P(X = x)$ for the probability that a random variable X is set to a specific value x. See Sect. 4.5 for basic probability notation.

Proposition 1. *The maximum likelihood estimator (MLE)*[2] *for* $P(\varphi(\pi))$ *where* $\pi = \langle e_1, \ldots, e_n \rangle$ *is*

$$\widehat{P(\varphi(\pi))} = \frac{1}{n} \sum_{i=1}^{n} \Lambda(e_i, \varphi). \tag{4.7}$$

◁

Proof. Since $X_{\varphi,t}$ *is a univariate Bernoulli random variable its MLE is well-established in the literature (see, e.g., [33]). Specifically, it equals to the ratio of 'successful trials' over the* n *trials.* ⊣

In order to obtain measures of interest for formulae and specifications we must, in addition, obtain estimators for the intersection of two LTL$_f$ formulae φ_1 and φ_2 being satisfied by a trace, e.g., $P(\varphi_1(\pi) \cap \varphi_2(\pi))$, and for the conditional distribution of φ_1 to be satisfied by trace π conditional on φ_2 being satisfied by the trace, e.g., $P(\varphi_1(\pi)|\varphi_2(\pi))$.

The latter will be particularly useful to extend the estimators to entire process specifications. Notice that we provide results for the satisfaction of formulae, yet similar results can be derived for violations by quantifying, e.g., $P(\neg\varphi_1(\pi) \cap \varphi_2(\pi))$ and $P(\neg\varphi_1(\pi)|\varphi_2(\pi))$. Formalizing the above, we wish to estimate the quantities of interest from a labeled sequence,

$$x_{(\varphi_1,\varphi_2),t} = \langle (\Lambda(e_i, \varphi_1), \Lambda(e_i, \varphi_2)) \rangle_{i=1}^{n},$$

for $\pi = \langle e_1, \ldots, e_n \rangle$.

Proposition 2. *The MLE for* $P(\varphi_1(\pi) \cap \varphi_2(\pi))$ *given a trace* $\pi = \langle e_1, \ldots, e_n \rangle$ *is*

$$\widehat{P(\varphi_1(\pi) \cap \varphi_2(\pi))} = \hat{p}_{11} = \frac{1}{n} \sum_{i=1}^{n} \Lambda(e_i, \varphi_1) \cdot \Lambda(e_i, \varphi_2). \tag{4.8}$$

◁

Proof. The proof follows from the MLE estimators for Bivariate Bernoulli random variables with corresponding success probabilities, p_{ij}*, with* $i, j \in \{0, 1\}$ *(see [106]).* ⊣

Having modeled the joint probability of two LTL$_f$ formulae satisfied by a trace, we can now define the probability of one formula being satisfied (or violated) by π conditioned on another formula being satisfied (or violated) by the same trace π. This result leads to the following estimator of the conditional probability.

Proposition 3. *The MLE for* $P(\varphi_1(\pi)|\varphi_2(\pi))$ *given a trace* $\pi = \langle e_1, \ldots, e_n \rangle$ *is*

$$\widehat{P(\varphi_1(\pi)|\varphi_2(\pi))} = \frac{\sum_{i=1}^{n} \Lambda(e_i, \varphi_1) \cdot \Lambda(e_i, \varphi_2)}{\sum_{i=1}^{n} \Lambda(e_i, \varphi_2)}. \tag{4.9}$$

◁

[2] MLE estimators exhibit important statistical properties such as unbiasedness and consistency (see Sect. 4.5.4).

Remark 1. When estimating $\widehat{P(\varphi_1(\pi)|\varphi_2(\pi))}$, the denominator of the estimator may be equal to 0. In such a case, the conditional probability is ill-defined and the trace is ignored for log-level computations; the value is denoted as NaN. ◁

4.7.2 Log estimators

We lift our results from traces to logs by estimating $P(\varphi(L))$, i.e., the probability that the log L satisfies a formula φ. Recall that an event log $L = \left\{\pi_1^{j_1}, \ldots, \pi_m^{j_m}\right\}$ is a bag of traces with trace $\pi_i^{j_i}$ occurring j_i times in the log. The size of the log $|L| = \sum_{i=1}^{m} j_i$ is the total number of traces.

Let us denote with $\bar{L} = \{\pi_1, \ldots, \pi_m\}$ the set of unique traces in L. We assume that the traces in L are independently generated by a trace generator π, which is, in turn, associated with a discrete probability function $P(T = \pi)$.[3] Let $\mathcal{T}$ be the support of the probability distribution of π, i.e.,

$$\mathcal{T} = \{\pi \mid P(T = \pi) > 0\}. \tag{4.10}$$

First, we generalize our definitions from a given trace π to a *random* trace π. To this end, we assume log completeness: the log contains all possible traces that can be generated from π, i.e., $\bar{L} = \mathcal{T}$. We plan to lift this assumption in future work.

Proposition 4. *The MLE for $P(\varphi(L))$ for a log L with a trace set $\bar{L} = \{t_i = \langle e_{i,1}, \ldots, e_{i,n_i}\rangle\}_{i=1}^{m}$ is*

$$\widehat{P(\varphi(L))} = \frac{1}{|L|} \sum_{i=1}^{m} \frac{j_i}{n_i} \sum_{k=1}^{n_i} \Lambda(e_{i,k}, \varphi)\,. \tag{4.11}$$

◁

with m being the number of unique traces in L, $|L|$ denoting the number of traces in the log, j_i the multiplicity of the i-th unique trace in L (so that $|L| = \sum_{i=1}^{m} j_i$), n_i the length of the i-th unique trace, and $e_{i,k}$ its k-th event.

To lift the estimators of intersection and conditional probabilities from traces to logs, we can again apply the law of total probability and derive the following.

Theorem 4. *The MLE of $P(\varphi_1(L) \cap \varphi_2(L))$ for a log L with a trace set $\bar{L} = \{\pi_i = \langle e_{i,1}, \ldots, e_{i,n_i}\rangle\}_{i=1}^{m}$ is*

$$\widehat{P(\varphi_1(L) \cap \varphi_2(L))} = \frac{1}{|L|} \sum_{i=1}^{m} \frac{j_i}{n_i} \sum_{k=1}^{n_i} \Lambda(e_{i,k}, \varphi_1)\,\Lambda(e_{i,k}, \varphi_2)\,. \tag{4.12}$$

◁

Lastly, we show an estimator for the conditional distribution $P(\varphi_1(L) \mid \varphi_2(L))$ – similarly, one can derive estimators for the other conditional probabilities as for traces.

[3] In practice, the trace can be generated via a random walk over, e.g., a finite-state automaton [66].

Theorem 5. *The MLE for $P(\varphi_1(L) \mid \varphi_2(L))$ for a log L with a trace set $\bar{L} = \{\pi_i = \langle e_{i,1}, \ldots, e_{i,n_i} \rangle\}_{i=1}^{m}$ is*

$$P(\varphi_1(\widehat{L) \mid \varphi_2(L)}) = \frac{\sum_{i=1}^{m} \frac{j_i}{n_i} \sum_{k=1}^{n_i} \Lambda(e_{i,k}, \varphi_1) \cdot \Lambda(e_{i,k}, \varphi_2)}{\sum_{i=1}^{m} \frac{j_i}{n_i} \sum_{k=1}^{n_i} \Lambda(e_{i,k}, \varphi_2)}. \tag{4.13}$$

◁

We conclude this section by highlighting that the computation of the estimators described thus far is tractable.

Theorem 6. *The estimators for LTL_f formulae being satisfied by a trace or a log, and the intersection and conditional probabilities thereof are computable in polynomial time.* ◁

Proof. The proof relies on Theorem 3 and Corollary 2: once we have checked and labeled the traces, the computation of estimators requires only queries over the resulting labels, which can be performed in $O(|L| \times n)$ considering n as the length of the longest trace in the log. ⊣

4.8 Evaluation and measurement of specifications

In the previous section, we estimated the probabilities of any LTL_f formula. Yet, as the evaluation of an RCon differs from that of an LTL_f formula (see Sect. 4.4), the evaluation of a specification consisting of RCons should take into account the interplay of activators and targets. The key point is in how to evaluate an *intersection* of RCons (i.e., a specification) on events. The rationale is that once we have the evaluation of the specification on every event, we can estimate the probabilities and consequently its measures like for any other RCon. In the remainder of this section, we start formalizing the semantics of RCon intersections in Sect. 4.8.1, continue proposing estimators of probabilities of traces and log satisfying such specifications (Sect. 4.8.2), describe the computation of interestingness measures (Sect. 4.8.3), and finally highlight advantages and practical implications of our approach (Sect. 4.8.4).

4.8.1 Evaluating specifications

Formally, we define the semantics of a specification $\mathcal{S}$ as follows: given a trace t of length n, an instant i with $1 \leq i \leq n$, and a specification $\mathcal{S} \triangleq \{\Psi_1, \ldots, \Psi_s\}$, with $s \in \mathbb{N}$ and $\Psi_j = \varphi_{\alpha_j} \boxdot\!\!\rightarrow \varphi_{\tau_j}$ for every j s.t. $1 \leq j \leq s$, we say that

- $\mathcal{S}$ is activated by π in i, i.e., $(\pi, i) \models \mathcal{S}_\alpha$, iff there exists a $\Psi_j \in \mathcal{S}$ s.t. $(\pi, i) \vDash \varphi_{\alpha_j}$;
- $\mathcal{S}$ is satisfied by π in i, $(\pi, i) \models \mathcal{S}$, iff $(\pi, i) \models \mathcal{S}_\alpha$ and there does not exist any $\Psi_j \in \mathcal{S}$ s.t. $(\pi, i) \not\models \Psi_j$;

- $\mathcal{S}$ is violated by π in i, $(\pi, i) \not\models \mathcal{S}$, iff there exists a $\Psi_j \in \mathcal{S}$ s.t. $(\pi, i) \not\models \Psi_j$;
- $\mathcal{S}$ is unaffected by π in i iff $(\pi, i) \not\models \mathcal{S}_\alpha$.

In other words, $\mathcal{S}$ is activated if at least one of its RCons is activated, satisfied if all and only its activated RCons are satisfied, violated if at least one activated RCon is violated, and unaffected if it is not activated.

In light of the above, we can express a specification $\mathcal{S} = \{\Psi_1, \ldots, \Psi_s\}$ as an RCon, $\mathcal{S} = \mathcal{S}_\alpha \boxdot\!\!\rightarrow \mathcal{S}_\tau$, where $\mathcal{S}_\alpha$ and $\mathcal{S}_\tau$ are LTL$_f$ formulae expressed as follows:

$$\mathcal{S}_\alpha = \bigvee_{j=1}^{s} \varphi_{\alpha_j}; \qquad \mathcal{S}_\tau = \bigwedge_{j=1}^{s} \neg\left(\varphi_{\alpha_j} \wedge \neg\varphi_{\tau_j}\right). \tag{4.14}$$

For example, $\mathcal{S}$ from Tab. 4.1 is activated when either Ψ_1 or Ψ_2 are (i.e., $\mathcal{S}_\alpha = \mathsf{c} \vee \mathsf{d}$) and it is satisfied when all the activated constraints are satisfied, i.e., $\mathcal{S}_\tau = (\neg\mathsf{c} \vee \Diamond\!\!\!\!-\,\mathsf{a}) \wedge (\neg\mathsf{d} \vee \Diamond\!\!\!\!-\,\mathsf{e})$. Hence, $\mathcal{S}$ is violated in $(\pi_3, 1)$, e.g, because Ψ_1 is violated and satisfied in $(\pi_3, 2)$, because Ψ_2 is satisfied and Ψ_1 is unaffected.

The possibility to reduce a specification to a single RCon is key to defining the corresponding estimators and thereby computing the probability a trace and a log satisfy it.

4.8.2 Estimators for Specifications

In this part, we define what trace and log estimators for single RCons, as well as for RCon specifications.

Trace Estimators

We first start by estimating the interestingness degree of an RCon. Let Λ_R be an RCon interpreter that takes an event and an RCon $\Psi = \varphi_\alpha \boxdot\!\!\rightarrow \varphi_\tau$ and returns a label $\Lambda_R(e, \Psi) \in \{0, \times, 1\}$ corresponding to Ψ being violated, unaffected, and satisfied by e, respectively. The labeling of an event given an RCon Ψ resorts to Λ (explained in Sect. 4.7.1) as follows:

$$\Lambda_R(e, \Psi) = \begin{cases} 0, & \text{if } \Lambda(e, \varphi_\alpha) = 1 \quad \text{and} \quad \Lambda(e, \varphi_\tau) = 0, \\ 1, & \text{if } \Lambda(e, \varphi_\alpha) = 1 \quad \text{and} \quad \Lambda(e, \varphi_\tau) = 1, \\ \times, & \text{otherwise.} \end{cases} \tag{4.15}$$

Notice that $\times$ is a new outcome of the labeling function Λ_R that solely applies to RCons but not to LTL$_f$ formulae (see the definition of Λ in Sect. 4.7.1). The second column of Tab. 4.1 lists the labels assigned by Λ_R to every event in the traces and all RCons in a sequence. For example, the evaluation of $\Psi_1 = \varphi_{\alpha_1} \boxdot\!\!\rightarrow \varphi_{\tau_1} = \mathsf{c} \boxdot\!\!\rightarrow \Diamond\!\!\!\!-\,\mathsf{a}$ on $\pi_4 = \langle e_{4,1}, \ldots, e_{4,6}\rangle = \langle \mathsf{b}, \mathsf{c}, \mathsf{a}, \mathsf{c}, \mathsf{e}, \mathsf{a}\rangle$ is such that $(\pi_4, 1) \not\models \mathsf{c}$, thus $\Lambda_R(e_{4,1}, \Psi_1) = \times$;

also, we observe that $(\pi_4, 2) \vDash \mathsf{c}$ but $(\pi_4, 2) \nvDash \Diamond\mathsf{a}$, therefore $\Lambda_R(e_{4,2}, \Psi_1) = 0$; finally, we have that $(\pi_4, 4) \vDash \mathsf{c}$ and $(\pi_4, 4) \vDash \Diamond\mathsf{a}$, so $\Lambda_R(e_{4,4}, \Psi_1) = 1$. Following this approach, we attain the following sequence of labels from π_4 via Λ_R on Ψ_1: $\langle \times, 0, \times, 1, \times, \times \rangle$.

We aim at estimating the probabilities of *interesting* satisfaction and violation of a given RCon in a trace, which correspond to cases in which the RCon is activated by it (see Sect. 4.4.1). Therefore, the corresponding random variable that describes an 'interesting' RCon (i.e., an RCon that is activated by a trace and satisfied in it), is

$$X_{\varphi_\tau, \pi} \mid X_{\varphi_\alpha, \pi} = 1.$$

As before, the latter is a univariate Bernoulli random variable with success probability of

$$P(\Psi(\pi)) = P(\varphi_\tau(\pi) \mid \varphi_\alpha(\pi)).$$

In other words, $P(\Psi(\pi))$ is the probability that an RCon Ψ is interestingly satisfied by some trace π. We are now ready to state our first estimation result.

Proposition 5. *The MLE of $P(\Psi(\pi))$ for a given trace $t = \langle e_1, \ldots, e_n \rangle$ is*

$$\widehat{P(\Psi(\pi))} = \frac{\sum_{i=1}^{n} \Lambda(e_i, \varphi_\tau(\pi)) \cdot \Lambda(e_i, \varphi_\alpha(\pi))}{\sum_{i=1}^{n} \Lambda(e_i, \varphi_\alpha(\pi))}. \tag{4.16}$$

◁

Proof. We need to estimate the probability that an RCon is interestingly satisfied by t,

$$P(\Psi(\pi)) = P(\varphi_\tau(\pi) \mid \varphi_\alpha(\pi)).$$

To this end, we can employ Proposition 3 to compute the corresponding MLE:

$$\widehat{P(\varphi_\tau(\pi) \mid \varphi_\alpha(\pi))} = \frac{\sum_{i=1}^{n} \Lambda(e_i, \varphi_\tau(\pi)) \cdot \Lambda(e_i, \varphi_\alpha(\pi))}{\sum_{i=1}^{n} \Lambda(e_i, \varphi_\alpha(\pi))}.$$

⊣

For example, $\widehat{P(\Psi_1(\pi_4))} = \frac{1}{2}$, as shown in Sect. 4.7.1. Moreover, note that for π_4, Ψ_2 is never activated, which leads to $\widehat{P(\Psi_2(\pi_4))} = \text{NaN}$. To obtain an MLE for $\widehat{P(\neg\Psi(\pi))}$ we can write

$$\widehat{P(\neg\Psi(\pi))} = 1 - \widehat{P(\Psi(\pi))}, \tag{4.17}$$

due to the result that a function of an MLE is an MLE [43].

At this stage, we turn to estimate the probabilities of interest for a specification $\mathcal{S}$ (i.e., a set of RCons). Specifically, since a specification is interpreted similarly to a single RCon, once we obtained the interpretation for $\mathcal{S}_\alpha$ and $\mathcal{S}_\tau$, we apply the

labeling mechanism Λ_R again to obtain one of the three possible outcomes:

$$\Lambda_R(e, \mathcal{S}) = \begin{cases} 0, & \text{if } \Lambda(e, \mathcal{S}_\alpha) = 1 \text{ and } \Lambda(e, \mathcal{S}_\tau) = 0, \\ 1, & \text{if } \Lambda(e, \mathcal{S}_\alpha) = 1 \text{ and } \Lambda(e, \mathcal{S}_\tau) = 1, \\ \times, & \text{otherwise.} \end{cases} \tag{4.18}$$

Note that φ_α and φ_τ are replaced by $\mathcal{S}_\alpha$ and $\mathcal{S}_\tau$, respectively, but the event-based labeling mechanism Λ_R remains unchanged. This allows for the re-use of Proposition 5 to obtain estimators for interesting satisfaction and violation of specifications denoted by $P(\mathcal{S}(\pi))$ and $P(\neg\mathcal{S}(\pi))$, respectively.

For estimating interesting satisfaction recall that $P(\mathcal{S}(\pi)) = P(\mathcal{S}_\tau(\pi) \mid \mathcal{S}_\alpha(\pi))$. Thus, we get the following result.

Proposition 6. *The MLE of $P(\mathcal{S}(\pi))$ for a given trace $t = \langle e_1, \ldots, e_n \rangle$ is*

$$\widehat{P(\mathcal{S}(\pi))} = \widehat{P(\mathcal{S}_\tau(\pi) \mid \mathcal{S}_\alpha(\pi))} = \frac{\sum_{i=1}^{n} \Lambda(e_i, \mathcal{S}_\tau) \cdot \Lambda(e_i, \mathcal{S}_\alpha)}{\sum_{i=1}^{n} \Lambda(e_i, \mathcal{S}_\alpha)}. \tag{4.19}$$

◁

Proof. Since $\mathcal{S}$ is an RCon, the MLE for the probability of interestingly satisfying it is given in Proposition 5. We plug-in the activator and target of the RCon, $\mathcal{S}_\alpha$ and $\mathcal{S}_\tau$ instead of φ_α and φ_τ, respectively, to complete the proof. ⊣

Notice that we use a similar notation for single RCons, replacing $\Psi(\pi)$ with $\mathcal{S}(\pi)$ to denote satisfaction of a specification. For example, $\widehat{P(\mathcal{S}(\pi_4))} = \frac{1}{2}$, as we consider only 0 or 1 outcomes applying the computation described in Proposition 6.

To obtain an MLE for $\widehat{P(\neg\mathcal{S}(\pi))}$ we can write

$$\widehat{P(\neg\mathcal{S}(\pi))} = 1 - \widehat{P(\mathcal{S}(\pi))}, \tag{4.20}$$

due to the result that a function of an MLE is an MLE [43].

Log Estimators

To derive log estimators we again assume that L is complete, i.e., $\bar{L} = \mathcal{T}$ with $\mathcal{T}$ being the support of the probability distribution of T. In what follows, we provide a result for a specification of RCons (including the special case that the specification is a single RCon). Let $P(\mathcal{S}(L))$ denote the probability of a log L to interestingly satisfy a specification $\mathcal{S}$.

Theorem 7. *The MLE of $P(\mathcal{S}(L))$ for a log L with a trace set $\bar{L} = \{\pi_i = \langle e_{i,1}, \ldots, e_{i,n_i} \rangle\}_{i=1}^{m}$ is given by*

$$\widehat{P(\mathcal{S}(L))} = P(\mathcal{S}_\tau(\widehat{L) \mid \mathcal{S}_\alpha}(L)) = \tag{4.21}$$
$$= \frac{\sum_{i=1}^{m} \frac{j_i}{n_i} \sum_{k=1}^{n_i} \Lambda(e_{i,k}, \mathcal{S}_\tau) \cdot \Lambda(e_{i,k}, \mathcal{S}_\alpha)}{\sum_{i=1}^{m} \frac{j_i}{n_i} \sum_{k=1}^{n_i} \Lambda(e_{i,k}, \mathcal{S}_\alpha)}. \quad \triangleleft$$

Proof. The proof follows from Theorem 5. Specifically, we replace φ_α and φ_τ in Eq. (4.13) with $\mathcal{S}_\alpha$ and $\mathcal{S}_\tau$, respectively. $\dashv$

Returning to the running example in Tab. 4.1, we have that $\widehat{P(\mathcal{S}(L))}$ is $\frac{(17{\cdot}0.44)+(6{\cdot}0.33)+(5{\cdot}0.40)+(12{\cdot}0.17)+(5{\cdot}0.0)}{(17{\cdot}0.44)+(6{\cdot}0.44)+(5{\cdot}0.50)+(12{\cdot}0.33)+(5{\cdot}0.0)} \approxeq 0.81$. To obtain an MLE for $\widehat{P(\neg\mathcal{S}(L))}$ we can write

$$\widehat{P(\neg\mathcal{S}(L))} = 1 - \widehat{P(\mathcal{S}(L))}, \tag{4.22}$$

due to the result that a function of an MLE is an MLE [43]. Notice that to obtain an estimator for a single RCon with activator φ_α and target φ_τ, we replace $\mathcal{S}_\alpha$ and $\mathcal{S}_\tau$ with φ_α and φ_τ, respectively.

Similarly to Sect. 4.7, we conclude by providing a result that states the computational complexity of our technique.

Theorem 8. *The estimation of the interestingness measures over specifications of RCon given an event log is polynomial.* $\triangleleft$

Proof. The proof relies on a similar argument to the proof of Theorem 6: each of the estimators is a query over the labeled sequences that can be computed in $O(|L| \times n)$, considering n as the length of the longest trace in L.

4.8.3 Computing Measures of Interestingness for Specifications

Having defined the estimators, we can now quantify the interestingness of rule-based LTL_f process specifications relying on association rule mining measures. Specifically, the estimates that we derive for specifications (i.e., Proposition 6 and Theorem 7) allow us to compute a plethora of interestingness measures while considering the joint effects of multiple rules at once.Thereby, we advance the state of the art as measures were previously restricted to a single-rule scope [44].

Table 4.3 shows the measures presented in Tab. 4.2 computed for a specification $\mathcal{S}$. In the following, we describe what these measures intuitively mean given a specification and an event log. As previously described, the event log serves as a means to estimate probabilities, which in turn are the building block of the interestingness measures. We define these measures on two levels. At a trace-level, they gauge interestingness of a specification based on the number of events that

Table 4.3. Interestingness measures from Tab. 4.2 defined for specifications

Measure	Trace	Log
Support	$P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau, t)$	$P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau, L)$
Confidence	$P(\mathcal{S}_\tau \mid \mathcal{S}_\alpha, t)$	$P(\mathcal{S}_\tau \mid \mathcal{S}_\alpha, L)$
Recall	$P(\mathcal{S}_\alpha \mid \mathcal{S}_\tau, t)$	$P(\mathcal{S}_\alpha \mid \mathcal{S}_\tau, L)$
Specificity	$P(\neg\mathcal{S}_\tau \mid \neg\mathcal{S}_\alpha, t)$	$P(\neg\mathcal{S}_\tau \mid \neg\mathcal{S}_\alpha, L)$
Lift	$\frac{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau, t)}{P(\mathcal{S}_\alpha, t)\,P(\mathcal{S}_\tau, t)}$	$\frac{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau, L)}{P(\mathcal{S}_\alpha, L)\,P(\mathcal{S}_\tau, L)}$

satisfy specific conditions. At a log-level, they take into account events *and* traces, which we shall collectively name as "fraction of the log" (or log fraction) for simplicity. Without loss of generality, we will describe the measures at a log-level. As it can be seen in Tab. 4.3, the same concepts seamlessly apply by restricting the notion of "log fraction" to that of "trace fraction" as the part of events in a single trace for the trace-level, due to the regularity of the definitions.

To determine the interestingness of a specification, Support considers the fraction of the log satisfying both its activator and its target. Confidence assesses in how far the target is satisfied within the log fraction that satisfies the activator. Dually, Recall reports on the events and traces that satisfy the activator among those that belong to the log fraction satisfying the target. Specificity quantifies the violation of the target within the fraction of the log that violates the activator. Finally, Lift scales Support by the product of the estimated probabilities of activator and target.

In the running example (see Tab. 4.1), we use Proposition 6 and Theorem 7 to compute the measures that appear in the last five columns (Support, Confidence, etc.) for $\mathcal{S} = \{\Psi_1, \Psi_2\}$. Observing the result in the last line of Tab. 4.1, e.g., we have that $\widehat{P(\mathcal{S}(L))}$ is 0.81, which together with the probabilities of activator and target of $\mathcal{S}$ yield a Support of 0.3 and Confidence of 0.81.

The measures used here are only an explanatory subset of the available ones. Thanks to the estimators presented throughout Sect. 4.8, it is possible to seamlessly compute all the ones presented in [44] and define new ones, based on the probabilities of activators and targets.

4.8.4 Discussion

Having laid down the theoretical foundations, we turn to discussing the proposed approach from a practical perspective. Intuitively, the probability estimators of an entire specification can be viewed as a relaxed metric of satisfaction, i.e., the extent to which the specification is satisfied in the given trace or log. It is a relaxed notion of satisfaction because it does not employ a strict boolean evaluation for the entire trace (as commonly done when evaluating LTL$_f$ formulae). The logical evaluation

is performed at the level of single events, while for traces and logs we use probability theory leading to relaxed statistical estimators. These estimators provide details about the severity of the observed violations in traces and logs.

Furthermore, log estimators are not biased by the length of the traces, but only by their frequency. Consider the following example: given a log of two traces, one composed of 100 events and the other of 10 events, suppose that every event in the first trace satisfies the specification, whereas every event of the second one violates it. Thus, 100 out of 110 events, namely about the 91 % of them, satisfy the specification. However, only 50 % of the traces satisfy it. A trace represents the execution of a process instance. Indeed, regardless of the number of steps required, the main information is the overall outcome from that instance, e.g., to which extent a specification holds true in that trace. This aspect is reflected in our log estimators: Given their definition in Sect. 4.8.2, it can be intuitively observed that the information of the trace used for the log aggregation is a fraction of events in the trace (i.e., those that satisfy a formula) and not their total number. Notice that the frequency of the trace in the log plays a major role, as it signals the probability of running the process in the way the trace reports.

The combination of rules in a specification is also a pivotal point of the contribution of our approach. Recall that a specification is not simply a container of independent rules, but a unique system composed of their simultaneous interactions. The classical approach to reason about specifications is to build a unique LTL_f formula formed by the logical conjunction of all the rules [125]. The disadvantage of such an approach is that the specification may be considered as violated by an entire trace for the violation of a rule by a single event. Since the specification is a unique system, it is agreeable that a violated rule implies the violation of the entire RCons specification, yet the finer granular level of the analysis confines this violation to a single event, and not to the entire trace. Therefore, an estimator for specifications should score low values in traces only if multiple events violate the rule. Extending the focus on event logs, the estimator should assign low values if numerous events in considerably many traces violate it.

Table 4.4 exemplifies the above reasoning. The specification $\mathcal{S} = \{\Psi_1, \ldots, \Psi_6\}$ is composed only of constraints formulated as $\mathit{True} \Box\!\!\rightarrow \neg z$, i.e., every event should be different from a given event z (with $z \in \{\mathtt{a}, \ldots, \mathtt{f}\}$). In the first trace, π_1, every single constraint is individually violated only by one event of the trace and satisfied by the others. Accordingly, their estimators assign high values (1.00). Nonetheless, the specification is violated in every single event by at least one constraint, thus the estimator's score for $\mathcal{S}$ is 0.00. Instead, every event of the second trace, π_2, satisfies Ψ_2, Ψ_3 and Ψ_4 (scoring 1.00), while Ψ_1, Ψ_5 and Ψ_6 are violated by one event therein

Table 4.4. Example of a specification violated by high probability constraints.

Log	Evaluation	RCon / Specification	P of RCon / P of $\mathcal{S}$
$\pi_1 = \langle \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f} \rangle$	$\langle 0, 1, 1, 1, 1, 1 \rangle$	$\Psi_1 = True \square\!\!\rightarrow \neg \mathsf{a}$	0.83
	$\langle 1, 0, 1, 1, 1, 1 \rangle$	$\Psi_2 = True \square\!\!\rightarrow \neg \mathsf{b}$	0.83
	$\langle 1, 1, 0, 1, 1, 1 \rangle$	$\Psi_3 = True \square\!\!\rightarrow \neg \mathsf{c}$	0.83
	$\langle 1, 1, 1, 0, 1, 1 \rangle$	$\Psi_4 = True \square\!\!\rightarrow \neg \mathsf{d}$	0.83
	$\langle 1, 1, 1, 1, 0, 1 \rangle$	$\Psi_5 = True \square\!\!\rightarrow \neg \mathsf{e}$	0.83
	$\langle 1, 1, 1, 1, 1, 0 \rangle$	$\Psi_6 = True \square\!\!\rightarrow \neg \mathsf{f}$	0.83
	$\langle 0, 0, 0, 0, 0, 0 \rangle$	$\mathcal{S} = \{\Psi_1, \ldots, \Psi_6\}$	0.00
$\pi_2 = \langle \mathsf{a}, \mathsf{y}, \mathsf{y}, \mathsf{y}, \mathsf{e}, \mathsf{f} \rangle$	$\langle 0, 1, 1, 1, 1, 1 \rangle$	$\Psi_1 = True \square\!\!\rightarrow \neg \mathsf{a}$	0.83
	$\langle 1, 1, 1, 1, 1, 1 \rangle$	$\Psi_2 = True \square\!\!\rightarrow \neg \mathsf{b}$	1.00
	$\langle 1, 1, 1, 1, 1, 1 \rangle$	$\Psi_3 = True \square\!\!\rightarrow \neg \mathsf{c}$	1.00
	$\langle 1, 1, 1, 1, 1, 1 \rangle$	$\Psi_4 = True \square\!\!\rightarrow \neg \mathsf{d}$	1.00
	$\langle 1, 1, 1, 1, 0, 1 \rangle$	$\Psi_5 = True \square\!\!\rightarrow \neg \mathsf{e}$	0.83
	$\langle 1, 1, 1, 1, 1, 0 \rangle$	$\Psi_6 = True \square\!\!\rightarrow \neg \mathsf{f}$	0.83
	$\langle 0, 1, 1, 1, 0, 0 \rangle$	$\mathcal{S} = \{\Psi_1, \ldots, \Psi_6\}$	0.50
$L = \{\pi_1^5, \pi_2^{10}\}$ $\lvert L \rvert = 15$		$\Psi_1 = True \square\!\!\rightarrow \neg \mathsf{a}$	0.83
		$\Psi_2 = True \square\!\!\rightarrow \neg \mathsf{b}$	0.94
		$\Psi_3 = True \square\!\!\rightarrow \neg \mathsf{c}$	0.94
		$\Psi_4 = True \square\!\!\rightarrow \neg \mathsf{d}$	0.94
		$\Psi_5 = True \square\!\!\rightarrow \neg \mathsf{e}$	0.83
		$\Psi_6 = True \square\!\!\rightarrow \neg \mathsf{f}$	0.83
		$\mathcal{S} = \{\Psi_1, \ldots, \Psi_6\}$	0.33

each (scoring 0.83). In this case, then, the estimator assigns 0.50 to the specification. Since the numerosity of π_1 in the event log is 5 and the numerosity of π_2 is twice as much (10), the specification on the entire log is assigned 0.33 by the estimator.

This is a desirable behavior: A specification is evaluated for the combination of its constraints, not its single parts. The first trace shows clearly that, despite the single constraints being mostly satisfied, together they do not properly capture the execution of the trace. Thus, the average of the measures would be a descriptive statistic for the single rules but a misleading indicator for the specification. Exploring extensions of weighted evaluations of specifications where specific constraints may be more relevant than others and treated unequally is an interesting future outlook.

4.9 Evaluation

The goal of this section is to demonstrate how gauging the interestingness of an entire process specification with multiple measures leads to novel results that are not achievable via the analysis of typical, single-rule based measures. To this end, we conduct three experiments. In Sect. 4.9.1, we show how the measure of a specification differs from the assemblage of the measures of single rules; in Sect. 4.9.2, we provide experimental evidence of the fact that distinct measures show different aspects of a specification, even in the case of full compliance with a Confidence level of 1.0; in Sect. 4.9.3, we propose a use-case application of our approach, analyzing its support to process drift detection. Finally, Sect. 4.9.4 concludes the section with a discussion

Table 4.5. Details of the real-world event logs used for the evaluation

Event log	Traces	Tasks	Events
BPIC12 [74]	13087	36	262200
BPIC13_cp [165]	1487	7	6660
BPIC14_f [75]	41353	9	369485
BPIC15_1f [76]	902	70	21656
RTFMP [124]	150370	11	561470
Sepsis [133]	1050	16	15214
Help-Desk [150]	4580	14	21348

of our findings.

We implemented our technique in a proof-of-concept Java tool, publicly available at https://github.com/Oneiroe/Janus. The implementation natively supports a relevant set of rule templates based on [81], but we already showed in Sect. 4.8 that the technique is seamlessly applicable to any RCon. Furthermore, inspired by [91], it supports the computation of 37 interestingness measures.

In our experiments, we used a set of openly available, real-world event logs. The details and references to the datasets are reported in Tab. 4.5. Note that our tool scales linearly with the size of the event log, the size of the specification, and the number of measures to compute. For example, for the Sepsis dataset the measurement took around 35 s with the heaviest setup (i.e., to compute 37 measures for a specification containing 3202 rules and a log with 1050 traces) on an Intel Core i5-7300U CPU at 2.60 GHz, quad-core, 16 GB of RAM, running Ubuntu 18.04.6.

4.9.1 Measuring single rules and entire specifications

In the following, we assess the usefulness of specifications measurement on real-life data, applying our technique to the results of various pattern-based LTL_f specification miners, i.e., Janus [45], MINERful [65], and Perracotta [175]. The goal is to highlight the importance of checking an entire specification, as opposed to the analysis of individual rules. The rationale is that while many specification miners use a threshold to retrieve only rules satisfied above that value, the corresponding specification may present a satisfaction degree below the desired level, following Morin's principle that the whole may be less than the sum of its parts [137].

The interested reader can find the scripts and input files to reproduce the tests alongside output reports and the full collection of specification rules at https://oneiroe.github.io/DeclarativeSpecificationMeasurements-Journal-static/. We conducted the experiment as follows. We discovered a specification from the log with each miner. Then, we used our tool to compute the interestingness measures on

the log. Here we focus on Confidence, as all miners implement a custom calculation for it. We repeated the discovery step with increasing Confidence thresholds, from 0 to 1, with step size of 0.05 [4]. Finally, we compared the measures of the specifications to the input threshold.

The results can be found in Figs. 4.1(a) to 4.1(g). As highlighted in Figs. 4.1(a), 4.1(d) and 4.1(e), every miner returned specifications whose overall Confidence is lower than the initially set threshold, even for thresholds greater that 0.7. This means that although every rule in a specification has a Confidence value greater than or equal to the threshold, the overall specification performs worse than the user-defined minimum accepted level.

Other datasets (see Figs. 4.1(b), 4.1(c), 4.1(f) and 4.1(g)), exhibit a similar behavior for lower thresholds.

This issue may lead to sub-optimal results, akin to the multiple testing problem [98] in statistical inference: assuming the independence of each hypothesis, thus not considering their inter-dependency, leads to erroneous outcomes. With our technique, this kind of issue becomes evident. Improving declarative process specification miners with the integration of our technique paves the path for interesting future endeavors. However, we remark that this analysis focuses on the quantitative matching of the specification to the logs, regardless of the semantics of the discovered specifications. Next, we broaden the perspective of interestingness from the sole Confidence to a larger set of measures.

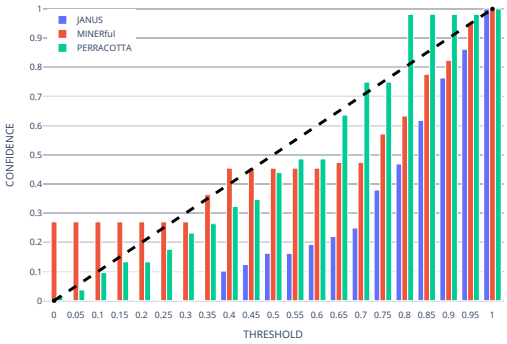
4.9.2 Differing measures

In this part, we show how different specifications, though never violated in the log, may exhibit different characteristics through the inspection of multiple measures.

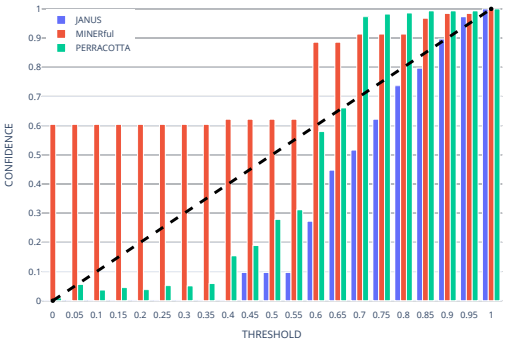
In this experiment, we make use of the Sepsis event log [133]. The dataset refers to a real-world process in the healthcare domain and contains records of patient visits with sepsis symptoms at a Dutch hospital. The dataset exhibits high variability: 75% of the traces are unique, which makes it a good candidate to evaluate partial specifications. To retrieve multiple distinct specifications, we first mined a specification with the miner set to discard partly violated rules. Any miner would be fit for the task, thus without loss of generality, we employed Janus with a Confidence threshold set to 1 for all rules. The resulting specification consisted of 238 rules, which we partitioned randomly into 5 subsets of RCons. Finally, we computed interestingness measures for each of the sub-specifications. An excerpt of the results is reported in Tab. 4.6.

The results show that despite a Confidence level of 1 for every specification (i.e.,

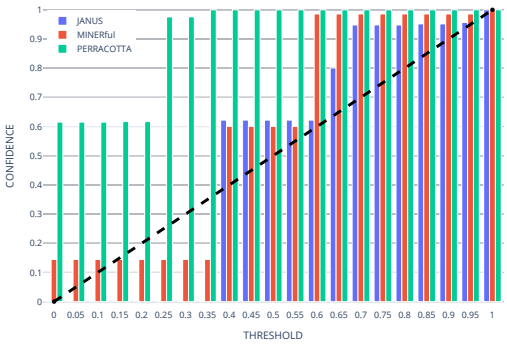
[4] For the BPIC 14 dataset Perracotta did not return any rules with thresholds above 0.55



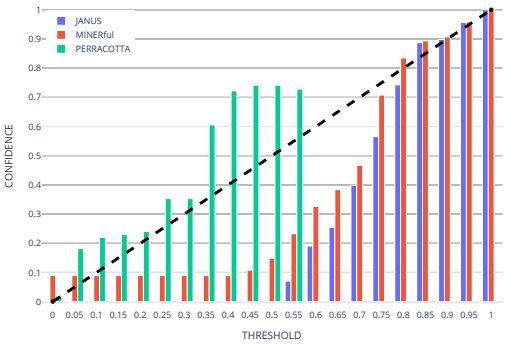
(a) Sepsis [133]



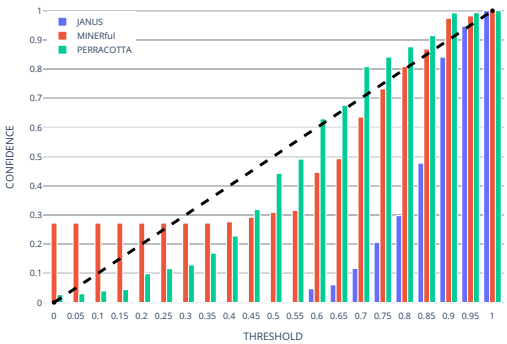
(b) BPIC12 [74]



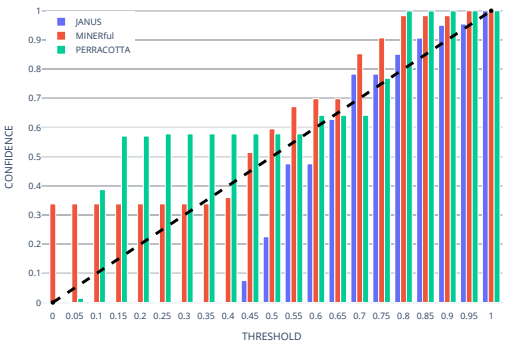
(c) BPIC13 [165]



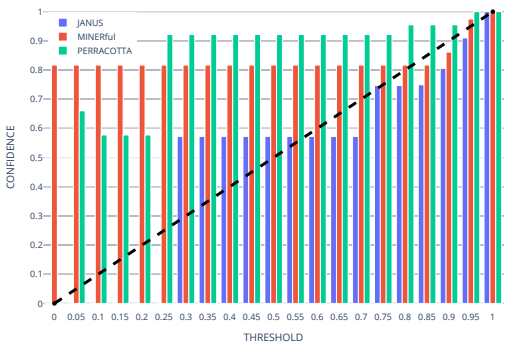
(d) BPIC14_f [75]



(e) BPIC15_1f [76]



(f) RTFMP [124]



(g) Help-Desk [150]

Figure 4.1. Confidence of the mined specifications with respect to the threshold used for their discovery

Table 4.6. Measurements of sub-specifications

Measure	Original $\mathcal{S}$	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$
Confidence	1.000	1.000	1.000	1.000	1.000	1.000
Support	1.000	0.003	0.011	0.074	0.522	0.683
Recall	1.000	0.011	1.000	1.000	1.000	1.000
Specificity	NaN	0.741	1.000	1.000	1.000	1.000
Lift	1.000	3.831	89.691	13.512	1.916	1.465

there are no violated rules), other measures can still detect differences. For example, Support shows that $\mathcal{S}_1$,$\mathcal{S}_2$, and $\mathcal{S}_3$ are applied to a very portion of events of the log, while $\mathcal{S}_4$ and $\mathcal{S}_5$ are activated by more than half of it. Specificity and Recall signal that for every specification the respective targets and activators always occur together, except for $\mathcal{S}_1$: its target occurs without the activator the vast majority of the time. The division by zero for the original $\mathcal{S}$ Specificity suggests that the specification was activated in every event of the log. Also, Lift shows that, in all the models, the joint probability of activator and target is higher than their individual one, yet with different strengths, e.g., $\mathcal{S}_2$ towering any other one. The input specifications and output measurements are available for reproducibility purposes at https://oneiroe.github.io/DeclarativeSpecificationMeasurements-Journal-static/.

To conclude, this experiment demonstrates the advantages of the availability of a full set of measures, going beyond the mere satisfaction of specifications. Indeed, the choice of the most appropriate measure will depend on the specific application of use, as we will exemplify next.

4.9.3 Impact of process drift on specification measures

Throughout this section, we analyze the impact that drifts [129] in process specifications may have on the various interestingness measures. Our objective is twofold. On the one hand, we aim to undertake a preliminary assessment of the capability of measures to reflect modifications in the process behavior. On the other hand, we want to observe the difference in sensitivity to changes exhibited by the measures originally suggested in the literature for association rule mining [168, 121] and adopted in the context of declarative process mining in [48]. Specifically, we perform the analysis for entire specifications and not only for individual rules.

We used the Visual Drift Detection (VDD) tool [179] as a benchmark to detect the points in which the process is subject to variations over time. VDD divides event logs into sub-logs of consecutive traces, measures the Confidence of all discoverable DECLARE constraints in each sub-log, and analyzes the time sequences generated by the Confidence measurements while grouping together constraints that exhibit

similar trends over time. It automatically detects change points within groups based on the oscillation of values in the sequence. Then, it automatically classifies those change points as either process drifts or evidence of erratic behavior and outliers [177].

Experimental setting

Taking the results illustrated in [177, 178] as a reference, we conducted our experiments with the following two real-world event logs: *(i)* the aforementioned Sepsis log [133], which reports on the tasks executed from the registration to the discharge of patients affected by sepsis, and *(ii)* the Help-Desk log [150], recording the activities carried out within a management process of the Help-desk of an Italian software company. The scripts we use for our tests can be found alongside the analysis results at the following address: https://github.com/l2brb/Measurement-change-point-evaluation.

To carry out our experiments, we went through the following steps. We first mined a process specification consisting of rules with high Confidence (rarely violated whenever triggered) from the event logs taken as a whole. To this end, we used Janus[5] [45] with the Confidence threshold set to 0.95. Then, we sliced the event log into consecutive sub-logs with a tumbling window approach, namely, extracting sequences of 50 consecutive, non-overlapping trace sets. We resorted to the dedicated pre-processing feature of MINERful[6] [65] to this extent. We fed the sub-logs into VDD for the detection of change points, setting the parameters as follows: *(i)* 50 for the window size, *(ii)* 50 for slide size, and *(iii)* 300 for the cut threshold. Finally, we made our tool compute the value of the measures of the initial specification on every sub-log in a sequence. Thereupon, we analyzed the trend of the specification measurements in correspondence with the change points detected by VDD, to observe if, and in how far, they exhibit variations. In the following, we discuss the results of our analysis.

Experimental results

Figures 4.2 and 4.3 illustrate the trends of Confidence (on the y-axis) for the mined specifications on the sub-logs of the Help-Desk and the Sepsis datasets, respectively. The points on the x-axis represent the timestamp of the first event in every sub-log. We add colored bars in correspondence with the change points detected by VDD. As Yeshchenko et al. explain in [177], two drifts occur in the course of the process

[5] https://github.com/Oneiroe/Janus/
[6] https://github.com/cdc08x/MINERful/

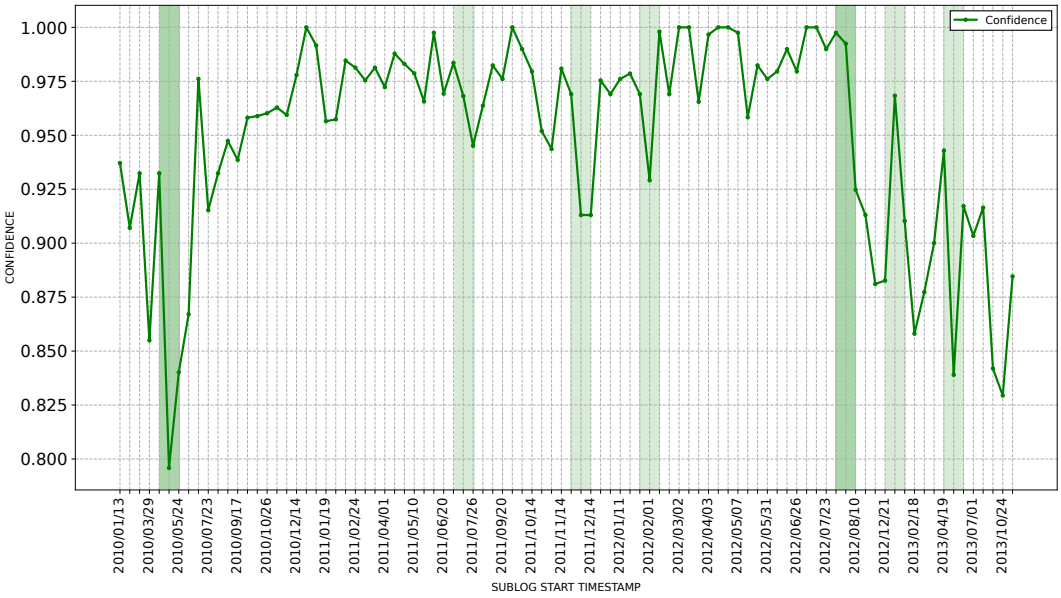


Figure 4.2. Help-Desk sub-log confidence values. The plot is visually edited with the addition of vertical bands in the areas where drifts (darker) and erratic behavior (lighter) are detected by VDD as in [177].

executions recorded in the Help-Desk log. They are located in the first and the last quarter of the diagram.

In Fig. 4.2, we highlight the corresponding points with a dark green bar. The lighter bars refer to the erratic behavior of the process in correspondence with other changing points of lower impact. We observe that all changing points correspond to upward or downward peaks in the poly-line plotting the Confidence values. However, they are not the sole steep slopes that can be noticed in the diagram. The reason is, VDD considers the trend of Confidence values for *all* discoverable constraints. In our case, instead, we focus on a selection of constraints, namely those that were discovered by Janus from the whole log setting a minimum threshold of 95 % for Confidence. Therefore, variations in the measure can be more visible considering the derived specification, while the oscillation may be mitigated by taking into account other constraints that are violated more often. Also, notice that VDD takes every constraint in isolation, while our holistic approach aims at measuring the interestingness of a specification as a whole.

Similar results are visible in Fig. 4.3. The bars in the diagram highlight the change points identified by VDD. As discussed in [178], none of the detected change points corresponds to actual process drifts. Instead, these oscillations are evidence of the erratic behavior characterizing the event log, most likely due to the reportedly flexible nature of the healthcare process involved [133]; moreover, the latest oscillation is an outlier that occurs towards the end of the event log. We observe that this

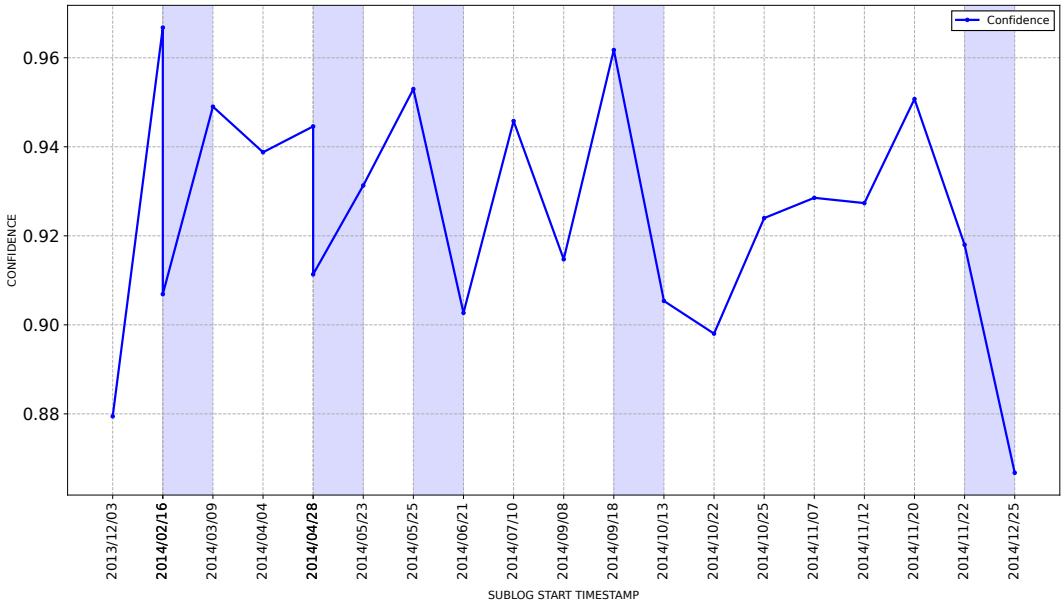


Figure 4.3. Sepsis sub-log confidence values. The plot is visually edited with the addition of vertical bands in the areas where erratic behavior is detected by VDD as in [178].

characteristic is also evidenced by the fact that seasonal oscillations characterize the entire diagram, though their amplitude is limited compared to the Help-Desk case.

Our approach allows one to gauge the interestingness of a specification for additional measures beyond Confidence. Specifically, Tab. 4.7 lists a comprehensive set of the measures that we used in our experiments. For every measure, we report the formula to compute it and its range.

The Sebag-Schoenauer measure of a specification $\mathcal{S}$ given an event log L, e.g., is computed as

$$1 - \frac{P(\mathcal{S}_\alpha, L)P(\neg\mathcal{S}_\tau, L)}{P(\mathcal{S}_\alpha \cap \neg\mathcal{S}_\tau, L)},$$

and its values range from 0 (included) to $+\infty$. Least Contradiction is defined as

$$\frac{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau, L) - P(\mathcal{S}_\alpha \cap \neg\mathcal{S}_\tau, L)}{P(\mathcal{S}_\tau, L)},$$

on the range $(-\infty, +\infty)$. Equipped with this set of measures, we conducted a comparative analysis to evaluate their ability to signal change points.

To allow for a comparison between measures defined on different ranges (e.g., $[0, +\infty)$ for the Sebag-Schoenauer measure, and $(-\infty, +\infty)$ for Least Contradiction), we normalize the values applying the method used in [48], which projects all values on a $[0, 1]$ interval (i.e., the range of Confidence, among others).

Tables 4.8(a) and 4.8(b) show the result of our analysis on both the Sepsis and Help-Desk logs, respectively. We report only measures with non-null values.

Table 4.7. Quality measures employed in the sensitivity experiment. For the sake of readability, we omit parameter L (the event log) from the formulae.

Measure	Formula	Range
Support	$P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau)$	$[0, 1]$
Confidence/ Precision	$P(\mathcal{S}_\tau \mid \mathcal{S}_\alpha)$	$[0, 1]$
Recall	$P(\mathcal{S}_\alpha \mid \mathcal{S}_\tau)$	$[0, 1]$
Accuracy	$P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau) + P(\neg\mathcal{S}_\alpha \cap \neg\mathcal{S}_\tau)$	$[0, 1]$
Lift/ Interest	$\frac{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau)}{P(\mathcal{S}_\alpha)P(\mathcal{S}_\tau)}$	$[0, +\infty)$
Leverage	$P(\mathcal{S}_\tau \mid \mathcal{S}_\alpha) - P(\mathcal{S}_\alpha)P(\mathcal{S}_\tau)$	$[-1, 1]$
Added Value	$P(\mathcal{S}_\tau \mid \mathcal{S}_\alpha) - P(\mathcal{S}_\tau)$	$[-1, 1]$
Jaccard	$\frac{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau)}{P(\mathcal{S}_\alpha) + P(\mathcal{S}_\tau) - P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau)}$	$[0, 1]$
Certainty factor	$\frac{P(\mathcal{S}_\tau \mid \mathcal{S}_\alpha) - P(\mathcal{S}_\tau)}{1 - P(\mathcal{S}_\tau)}$	$[-1, 1]$
Klosgen	$\sqrt{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau)} \times \max\left(P(\mathcal{S}_\tau \mid \mathcal{S}_\alpha) - P(\mathcal{S}_\tau), P(\mathcal{S}_\alpha \mid \mathcal{S}_\tau) - P(\mathcal{S}_\alpha)\right)$	$[-1, 1]$
Conviction	$\frac{P(\mathcal{S}_\alpha)P(\neg\mathcal{S}_\tau)}{P(\mathcal{S}_\alpha \cap \neg\mathcal{S}_\tau)}$	$[0, +\infty)$
J-Measure	$P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau) \log \frac{P(\mathcal{S}_\tau \mid \mathcal{S}_\alpha)}{P(\mathcal{S}_\tau)} + P(\mathcal{S}_\alpha \cap \neg\mathcal{S}_\tau) \log \frac{P(\neg\mathcal{S}_\tau \mid \mathcal{S}_\alpha)}{P(\neg\mathcal{S}_\tau)}$	$(-\infty, +\infty)$
One-Way Support	$P(\mathcal{S}_\tau \mid \mathcal{S}_\alpha) \log_2 \frac{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau)}{P(\mathcal{S}_\alpha)P(\mathcal{S}_\tau)}$	$(-\infty, +\infty)$
Two-Way Support	$P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau) \log_2 \frac{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau)}{P(\mathcal{S}_\alpha)P(\mathcal{S}_\tau)}$	$(-\infty, +\infty)$
Piatetsky-Shapiro	$P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau) - P(\mathcal{S}_\alpha)P(\mathcal{S}_\tau)$	$[-1, 1]$
Cosine	$\frac{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau)}{\sqrt{P(\mathcal{S}_\alpha)P(\mathcal{S}_\tau)}}$	$[0, +\infty)$
Loevinger	$1 - \frac{P(\mathcal{S}_\alpha)P(\neg\mathcal{S}_\tau)}{P(\mathcal{S}_\alpha \cap \neg\mathcal{S}_\tau)}$	$(-\infty, 1]$
Information Gain	$\log \frac{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau)}{P(\mathcal{S}_\alpha)P(\mathcal{S}_\tau)}$	$(-\infty, +\infty)$
Sebag-Schoenauer	$\frac{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau)}{P(\mathcal{S}_\alpha \cap \neg\mathcal{S}_\tau)}$	$[0, +\infty)$
Least Contradiction	$\frac{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau) - P(\mathcal{S}_\alpha \cap \neg\mathcal{S}_\tau)}{P(\mathcal{S}_\tau)}$	$(-\infty, +\infty)$
Odd Multiplier	$\frac{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau)P(\neg\mathcal{S}_\tau)}{P(\mathcal{S}_\tau)P(\mathcal{S}_\alpha \cap \neg\mathcal{S}_\tau)}$	$[0, +\infty)$
Example and Counterexample Rate	$1 - \frac{P(\mathcal{S}_\alpha \cap \neg\mathcal{S}_\tau)}{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau)}$	$(-\infty, 1]$
Zhang	$\frac{P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau) - P(\mathcal{S}_\alpha)P(\mathcal{S}_\tau)}{\max\left(P(\mathcal{S}_\alpha \cap \mathcal{S}_\tau)P(\neg\mathcal{S}_\tau), P(\mathcal{S}_\tau)P(\mathcal{S}_\alpha \cap \neg\mathcal{S}_\tau)\right)}$	$(-\infty, +\infty)$

Table 4.8. Variability of specification measures in the context of drift detection.

(a) Sepsis [133]

Measure	Mean	Std. Dev.	CV
Sebag-Schoenauer	0.861,505,846,320	0.002,053,050,180	0.002,383,094,890
Compliance	0.924,989,480,532	0.000,696,413,129	0.000,752,887,620
Jaccard	0.924,989,494,280	0.000,696,413,103	0.000,752,887,582
Accuracy	0.924,989,494,150	0.000,696,412,945	0.000,752,887,410
Support	0.924,989,494,150	0.000,696,412,945	0.000,752,887,410
Confidence	0.924,989,487,357	0.000,696,412,870	0.000,752,887,335
Cosine	0.962,422,081,608	0.000,175,943,255	0.000,182,812,987
Example and Counterexample Rate	0.969,838,405,557	0.000,159,563,237	0.000,164,525,591
Least Contradiction	0.727,378,805,369	$8.975{,}428{,}878{,}985 \times 10^{-5}$	$1.233{,}941{,}491{,}385 \times 10^{-4}$
Certainty factor	0.499,999,875,531	$3.627{,}784{,}544{,}022 \times 10^{-13}$	$7.255{,}570{,}894{,}236 \times 10^{-13}$
Zhang	0.499,999,929,596	$1.143{,}870{,}607{,}449 \times 10^{-13}$	$2.287{,}741{,}537{,}031 \times 10^{-13}$
Leverage	0.499,999,971,617	$1.833{,}876{,}167{,}754 \times 10^{-14}$	$3.667{,}752{,}543{,}712 \times 10^{-14}$
Added Value	0.499,999,985,808	$4.584{,}691{,}433{,}807 \times 10^{-15}$	$9.169{,}383{,}127{,}870 \times 10^{-15}$
Piatetsky-Shapiro	0.499,999,985,808	$4.584{,}691{,}433{,}807 \times 10^{-15}$	$9.169{,}383{,}127{,}870 \times 10^{-15}$
Klosgen	0.499,999,986,751	$3.992{,}108{,}862{,}561 \times 10^{-15}$	$7.984{,}217{,}936{,}682 \times 10^{-15}$
Two-way Support	0.499,999,994,921	$2.812{,}996{,}585{,}923 \times 10^{-15}$	$5.625{,}993{,}228{,}996 \times 10^{-15}$
One-way Support	0.499,999,994,921	$2.812{,}996{,}238{,}645 \times 10^{-15}$	$5.625{,}992{,}534{,}439 \times 10^{-15}$
Lovinger	0.666,666,658,768	$2.571{,}487{,}448{,}916 \times 10^{-15}$	$3.857{,}231{,}219{,}076 \times 10^{-15}$
Information Gain	0.499,999,995,743	$1.801{,}732{,}986{,}214 \times 10^{-15}$	$3.603{,}466{,}003{,}112 \times 10^{-15}$
J Measure	0.499,999,996,042	$1.750{,}513{,}976{,}295 \times 10^{-15}$	$3.501{,}027{,}980{,}303 \times 10^{-15}$
Lift	0.500,000,002,619	$3.690{,}476{,}483{,}444 \times 10^{-16}$	$7.380{,}952{,}928{,}225 \times 10^{-16}$
Conviction	0.500,000,006,667	$2.583{,}333{,}354{,}669 \times 10^{-16}$	$5.166{,}666{,}640{,}448 \times 10^{-16}$
Odd Multiplier	0.500,000,005,000	$5.176{,}899{,}690{,}513 \times 10^{-32}$	$1.035{,}379{,}927{,}749 \times 10^{-31}$
Recall	1.000,000,000,000	0.000,000,000,000	0.000,000,000,000

(b) Help-Desk [150]

Measure	Mean	Std. Dev.	CV
Sebag-Schoenauer	0.900,819,237,282	0.006,297,508,246	0.006,990,867,852
Accuracy	0.950,622,074,373	0.002,179,473,562	0.002,292,681,414
Support	0.950,622,074,373	0.002,179,473,562	0.002,292,681,414
Compliance	0.950,622,071,166	0.002,179,473,472	0.002,292,681,328
Confidence	0.950,622,076,105	0.002,179,473,334	0.002,292,681,170
Jaccard	0.950,622,081,188	0.002,179,473,205	0.002,292,681,022
Cosine	0.975,239,803,262	0.000,551,582,239	0.000,565,586,266
Example and Counterexample Rate	0.979,413,924,601	0.000,524,304,351	0.000,535,324,584
Least Contradiction	0.734,560,445,157	0.000,294,921,192	0.000,401,493,430
Certainty factor	0.500,000,047,986	$3.021{,}126{,}604{,}882 \times 10^{-13}$	$6.042{,}252{,}629{,}875 \times 10^{-13}$
Zhang	0.500,000,026,811	$8.434{,}163{,}170{,}167 \times 10^{-14}$	$1.686{,}832{,}543{,}582 \times 10^{-13}$
Leverage	0.500,000,009,070	$4.133{,}232{,}125{,}709 \times 10^{-15}$	$8.266{,}464{,}101{,}459 \times 10^{-15}$
One-way Support	0.500,000,008,712	$1.037{,}287{,}060{,}395 \times 10^{-15}$	$2.074{,}574{,}084{,}643 \times 10^{-15}$
Two-way Support	0.500,000,008,712	$1.037{,}287{,}000{,}451 \times 10^{-15}$	$2.074{,}573{,}964{,}754 \times 10^{-15}$
Added Value	0.500,000,004,535	$1.033{,}308{,}212{,}580 \times 10^{-15}$	$2.066{,}616{,}406{,}415 \times 10^{-15}$
Piatetsky-Shapiro	0.500,000,004,535	$1.033{,}308{,}212{,}580 \times 10^{-15}$	$2.066{,}616{,}406{,}415 \times 10^{-15}$
Klosgen	0.500,000,004,274	$9.205{,}376{,}653{,}863 \times 10^{-16}$	$1.841{,}075{,}315{,}036 \times 10^{-15}$
J Measure	0.500,000,007,383	$6.863{,}433{,}069{,}274 \times 10^{-16}$	$1.372{,}686{,}593{,}585 \times 10^{-15}$
Information Gain	0.500,000,006,803	$6.266{,}684{,}578{,}018 \times 10^{-16}$	$1.253{,}336{,}898{,}551 \times 10^{-15}$
Lovinger	0.666,666,669,648	$8.286{,}719{,}455{,}800 \times 10^{-16}$	$1.243{,}007{,}912{,}811 \times 10^{-15}$
Lift	0.500,000,007,826	$1.111{,}562{,}317{,}104 \times 10^{-16}$	$2.223{,}124{,}599{,}412 \times 10^{-16}$
Conviction	0.500,000,003,095	$8.849{,}684{,}779{,}794 \times 10^{-17}$	$1.769{,}936{,}945{,}002 \times 10^{-16}$
Odd Multiplier	0.500,000,005,000	$1.995{,}913{,}133{,}692 \times 10^{-31}$	$3.991{,}826{,}227{,}465 \times 10^{-31}$
Recall	1.000,000,000,000	0.000,000,000,000	0.000,000,000,000

For every measure, we indicate the mean value, the standard deviation, and the Coefficient of Variation (CV). CV, expressed as the ratio of the standard deviation to the mean, shows the extent of variability in relation to the average value along the sequence of collected values. We sort the measures in descending order by CV, standard deviation and mean. Measures that exhibit relatively more ample oscillations (hence, most sensitive to changes) are thus at the top. Both with the Sepsis and Help-Desk logs, the Sebag-Schoenauer measure ranks first, followed by a group of measures including Support and Confidence with very close values. From the Least Contradiction on, all measures follow with a CV that is lower by orders of magnitude (from 10^{-4} down to 10^{-13} and below). The Recall measure closes the list with a steady mean of 1.0, as both CV and standard deviation equate to 0. Notice that, besides minimal variations in the ranking, the aforementioned groups are composed by the same measures both in Tabs. 4.8(a) and 4.8(b), thus regardless of the event log under examination.

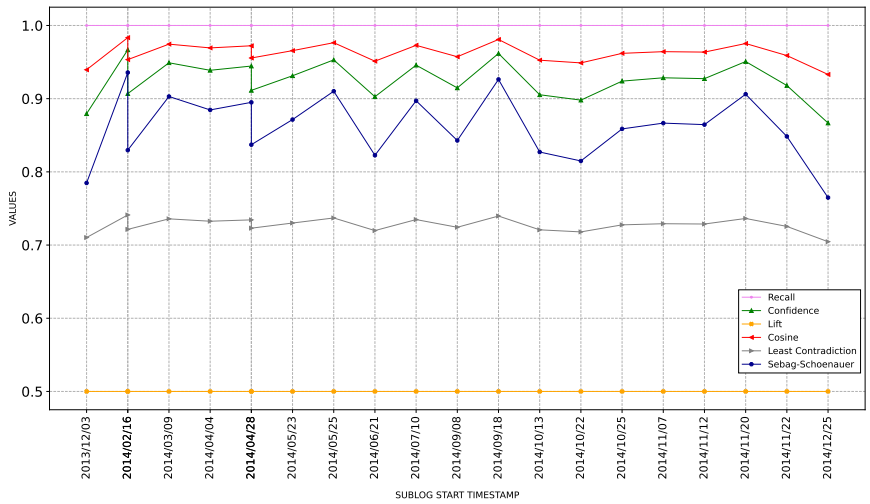


Figure 4.4. Specification measure oscillations on the Sepsis event log

Figures 4.4 and 4.5 plot the trends of those measures to visually compare their variations with the sub-logs extracted from the Sepsis and Help-Desk datasets, respectively. In both cases, we observe that the amplest oscillations are exhibited by the Sebag-Schoenauer measure, as expected in light of the previous discussion. Similar trends characterize the polylines associated with measures representing the group at the top of Tabs. 4.8(a) and 4.8(b), such as Confidence and Cosine. Least Contradiction lies in the middle of the diagrams and, though subject to variations, shows a lower sensitivity to changes. Lift is a representative of a large group of

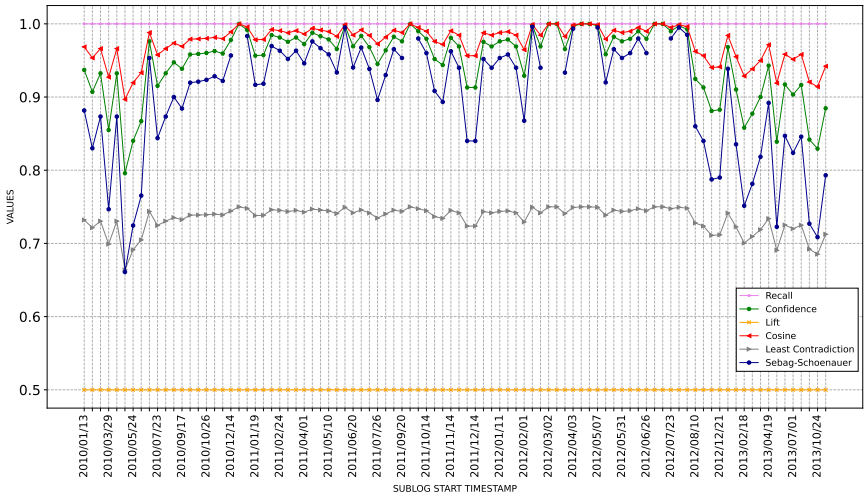


Figure 4.5. Specification measure oscillations on the Help-Desk event log

measures showing only imperceptible fluctuations. Finally, as anticipated above, Recall remains steadily equal to 1.0.

The trends exhibited in Figs. 4.2 to 4.5 pertain to full specifications mined from the original event logs. Our approach, however, can gauge the interestingness of any set of constraints, including sub-specifications (as illustrated in the experiment on synthetic data in Sect. 4.9.1). Here, we observe the variations that Confidence undergoes for some sub-specifications in particular. To have a better understanding of the rules that have led to the drifts, we form these sub-specifications by joining the constraints in the clusters that exhibit the most erratic trends in the Help-Desk log [150] and the Sepsis log [150] according to [177] and [178], respectively. The authors of [177, 178] indicate that those clusters are the ones that mainly signal change points in the process. Here, we leverage the capability of our approach to assess the behavior of those constraints taken together as sub-specifications, rather than individually as in [177, 178]. Thereafter, we also create specifications that join those sub-specifications. Finally, we measure Confidence for every sub-specification and specification on the respective sub-logs, using the same tumbling window approach as in the experiments above.

Tables 4.9 and 4.10 report on the minimum, maximum, and average values of Confidence together with the standard deviation and coefficient of variation for the clusters and the specifications derived therefrom on the Help-Desk and Sepsis event logs, respectively. We can observe that the erraticity of the clusters' behavior is confirmed by the CV, which is orders of magnitude higher than the values reported

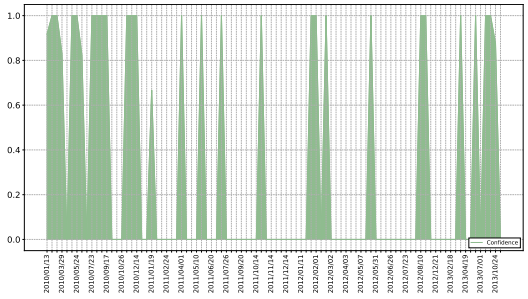
Table 4.9. Aggregate confidence in the Help-Desk sub-logs [150] for the constraint clusters exhibiting the most erratic behavior [177]

Cluster	Mean	Std. Dev	CV	Max	Min
Cluster 4	0.6336	0.0083	0.0131	0.8912	0.5129
Cluster 9	0.3164	0.2110	0.6670	1.0000	0.0000
Cluster 11	0.0543	0.0520	0.9560	1.0000	0.0000
Joint specification	0.6346	0.0082	0.0128	0.8918	0.5129

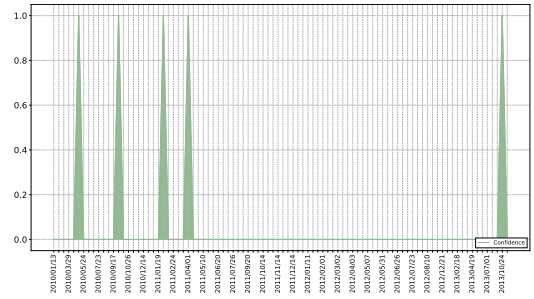
in Tabs. 4.8(a) and 4.8(b). Figures 4.6 and 4.7 illustrate the trend of Confidence over the subsequent sub-logs from the Help-Desk and Sepsis datasets, respectively. Both figures consider the individual clusters alone (Figs. 4.6(a) to 4.6(c), 4.7(a) and 4.7(b)) and the joined specifications (Figs. 4.6(d) and 4.7(c)). The plots in Figs. 4.6(a) to 4.6(c), 4.7(a) and 4.7(b), pertaining to the individual sub-specifications, closely resemble the trends illustrated in [177, 178]. However, we can also observe particular phenomena due to the definition of Confidence itself (see Tab. 4.7): *(i)* The Confidence of joined specifications tends to take the non-zero values from the sub-specifications, on one hand; *(ii)* On the other hand, the height of the peaks stemming from some sub-specifications is reduced whenever in other sub-specifications a corresponding lower Confidence is reported.

The former effect is particularly visible comparing Figs. 4.7(a) and 4.7(b) with Fig. 4.7(c): the plateaus lying at 0 occur in Fig. 4.7(c) only in correspondence of 0 values for the Confidence of *both* the clusters the specification is composed of. Such a plateau, e.g., occurs between the marks "2014/03/09" and "2014/12/25" in Fig. 4.7(b) (cluster 12), but not in Fig. 4.7(a) (cluster 8). Then, the plateau does not occur in the Confidence plot for the whole specification in Fig. 4.7(c). We see plateaus of Confidence equal to 0 in Fig. 4.7(c) right before the "2014/03/09" and "2014/11/07", instead, as the Confidence of neither of the sub-specifications goes beyond the minimum.

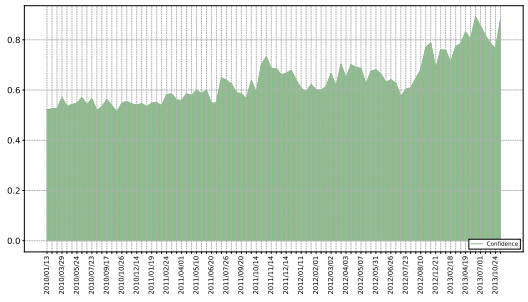
The conjunction of the two aforementioned effects is instead noticeable from Figs. 4.6(c) and 4.6(d). Confidence for cluster 4 (Fig. 4.6(c)) neither drops to 0 nor raises to 1 with any sub-log, as opposed to clusters 9 and 11 (Figs. 4.6(a) and 4.6(b); see also the minimum and maximum values reported in Tab. 4.8(b)). Then, the plot in Fig. 4.6(d) tends to almost completely overlap with that of Fig. 4.6(c). We remark that this effect differs from what we would have obtained by merely averaging the Confidence values of the clusters' sub-specifications to assign the Confidence values of the joined specification.



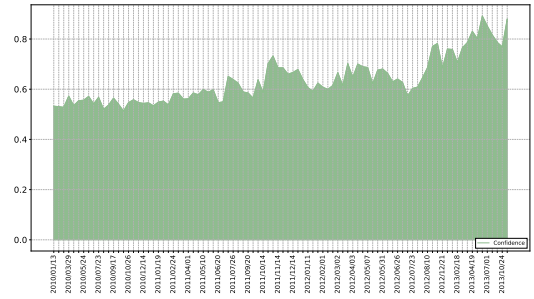
(a) Cluster 9



(b) Cluster 11

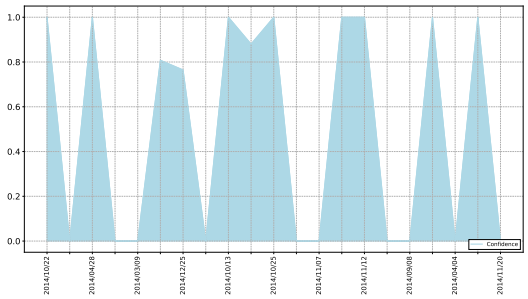


(c) Cluster 4

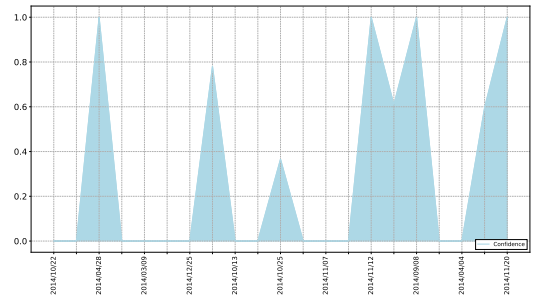


(d) Specification consisting of the constraints in the clusters

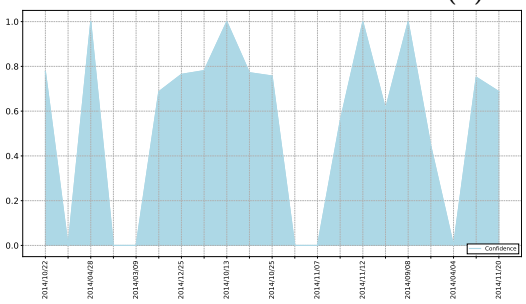
Figure 4.6. Confidence of the constraint clusters with the most erratic behavior with the Help-Desk log, and of the specification stemming from their union



(a) Cluster 8



(b) Cluster 12



(c) Specification consisting of the constraints in the clusters

Figure 4.7. Confidence of the constraint clusters with the most erratic behavior with the Sepsis log, and of the specification stemming from their union

Table 4.10. Aggregate confidence in the Sepsis sub-logs [133] for the constraint clusters exhibiting the most erratic behavior [178]

Cluster	Mean	Std. Dev	CV	Max	Min
Cluster 8	0.4977	0.2404	0.4831	1.0000	0.0000
Cluster 12	0.3025	0.1775	0.5866	1.0000	0.0000
Joint specification	0.5518	0.1481	0.2683	1.0000	0.0000

4.9.4 Discussion

Thus far, we have investigated various aspects to which the application of our measurement framework contributes with novel insights. We provide their summary below.

First of all, we observed in Sect. 4.9.1 that the overall compliance of an event log to a whole specification tends to be lower than its compliance to every rule taken individually. This aspect is of considerable relevance in the context of process and specification mining, as current works in the literature still tend to resort to an analysis centered around individual rules, thus neglecting their effect on the expected behavior in combination [175, 120, 116, 65, 17, 24].

Also, the opportunity to compute a large array of measures, including but not limited to those that derive from association rule mining, sheds light on interesting facts from different perspectives. The empirical piece of evidence presented in Sect. 4.9.2 confirms that a single measure is not sufficient to provide a full account of the degree of interestingness of a mined process specification. The use case illustrated in Sect. 4.9.3 shows a possible application of our approach to assessing the variability of the degree of compliance of process executions with the overall specification over time. Our tool is not meant to be a drift identification technique. Other approaches, such as the aforementioned VDD [178], are specifically designed to achieve this goal. However, our method can be used in practice to observe the effect that drifts and other change points in the process may have on an entire specification [161]. The most suitable measure to consider for this kind of analysis appears to be the Least Contradiction. Also, the oscillations of measures in the sequence highlight that process executions are subject to variations over time, while the mining of a whole event log cannot represent these changes in behavior (either temporary or not). Furthermore, we have observed that the mere aggregation of measures stemming from sub-specifications does not properly account for the measurement of a conjoint specification.

Chapter 5

Decentralized Conformance Checking from Different Data Sources

5.1 Introduction

In today's business landscape, organizations constantly seek ways to enhance operational efficiency, increase performance, and gain insights to improve their processes. Process mining offers techniques to discover, monitor, and improve business processes by extracting knowledge from chronological records recorded by process-aware information systems, i.e., the *event logs* [63]. The vast majority of process mining contributions consider *intra-organizational* settings, in which processes are executed inside individual organizations. However, organizations increasingly recognize the value of collaboration and synergy in achieving operational excellence. *Inter-organizational* business processes involve several independent organizations cooperating to achieve a shared objective. Process mining can bring the advantages of transparency, performance optimization, and benchmarking in this context [5]. Since different process data owners feed separate mining nodes, this setting characterizes what we call *decentralized process mining*.

Among the various process mining tasks, conformance checking (see Sect. 2.2.3) assesses how observed executions adhere to a given process representation and, in decentralized settings, enables cross-organizational monitoring and benchmarking of a shared process while preserving local operational autonomy. Inter-organizational processes are rarely governed by a single regulation since each party may impose its own compliance obligations, privacy constraints, and operational policies. Declarative specifications naturally fit this context, as they provide modular sets of constraints

that can be checked locally and, when permitted, composed into an overall specification for joint analysis. This aligns with the focus of the present dissertation on declarative process representations. Notably, the challenge of decentralized execution data is not restricted to declarative process mining. More generally, process mining algorithms, regardless of whether they target declarative or imperative representations, require access to process execution data that may be fragmented across independent organizations. Building on the findings of Chapter 4, conformance checking for declarative specifications should be performed at the level of the overall specification, since interactions among rules cannot be captured by aggregating rule-level or local analyses. When execution data is distributed across independent organizations, obtaining this holistic view from fragmented logs becomes particularly challenging and raises confidentiality concerns.

Companies, though, are reluctant to share private information required to execute algorithms with external parties [127], thus hindering its adoption. Letting sensitive operational data traverse organizational boundaries introduces concerns about data secrecy, security, and compliance with internal regulations [138].

To address this issue, the majority of research endeavors have focused thus far on the alteration of input data or of intermediate analysis by-products, with the aim to impede the counterparts from reconstructing the original information sources [86, 85, 83, 82]. These preemptive solutions have the remarkable merit of neutralizing information leakage by malicious parties a priori. Nevertheless, they entail an ex-ante information loss, thus compromising downstream process mining capabilities [86, 85], or require the execution of computationally heavy protocols undermining scalability [83, 82].

These limitations call for approaches that enable executing conformance checking and mining algorithms in decentralized settings without sacrificing confidentiality, which motivates our third research question:

> **RQ3: Decentralized Data Sources**
>
> How can declarative specifications be checked against execution data that is distributed across multiple independent sources?

To answer RQ3 and overcome these limitations, we propose CONFINE, a novel approach and tool aimed at enhancing collaborative information system architectures with secrecy-preserving process mining capabilities. CONFINE is model-agnostic and can support process mining tasks involving both declarative and imperative process representations, including process discovery, conformance checking, and related analyses. To secure information secrecy during the exchange and elaboration of data, our solution resorts to *Trusted Execution Environments* (TEEs) [159], namely

hardware-secured contexts that guarantee code integrity and data confidentiality before, during, and after their utilization. Owing to these characteristics, CONFINE lets information be securely transferred beyond the organizations' borders. Therefore, computing nodes other than the information provisioners can aggregate and elaborate the original, unaltered process data in a secure, externally inaccessible vault. Also, CONFINE is capable of providing these guarantees while demanding scalable computational overhead.

The decentralized architecture of CONFINE supports a four-staged protocol:

(i) The initial exchange of preliminary metadata, *(ii)* the attestation of the mining entities, *(iii)* the secure transmission and secrecy-preserving merge of encrypted information segments amid multiple parties, *(iv)* the isolated and verifiable computation of process discovery algorithms on joined data. We evaluate our proof-of-concept implementation against synthetic and real-world data with a convergence test followed by experiments to assess the scalability of our approach. Since TEEs operate with dedicated memory pages shielded from access by external entities (operating system included), thus entailing a hardware constraint on computation space, we endow our experiments with an analysis of memory usage, too.

5.2 Related Work

The scientific literature already includes noticeable contributions to process mining in a decentralized setting with a focus on data secrecy, despite the relative recency of this research branch across process mining and collaborative information systems. The work of Müller et al. [139] revolves around data privacy and security within third-party systems that mine data generated from external providers on demand. To safeguard the integrity of data earmarked for mining purposes, their research introduces a conceptual architecture that entails the execution of process mining algorithms within a cloud service environment, fortified with Trusted Execution Environments. Drawing inspiration from this foundational contribution, our research work seeks to design a decentralized approach characterized by organizational autonomy in the execution of process mining algorithms, devoid of synchronization mechanisms taking place between the involved parties. A notable departure from the framework of Müller et al. lies in the fact that here each participating organization retains the discretion to choose when and how mining operations are conducted. Moreover, we bypass the idea of fixed roles, engineering a peer-to-peer scenario in which organizations can simultaneously be data provisioners or miners. Fahrenkrog-Petersen et al. [86, 85] theorize the PRETSA algorithms family, namely a set of event log sanitization techniques that perform step-wise transformations of prefix-tree event

log representation into a sanitized output ensuring *k-anonimization* and *t-closeness*. While these algorithms effectively minimize information loss, they introduce targeted approximations within the original event log, which may compromise the exactness of process mining results or inhibit mining tasks. In contrast, our research proposes an architecture wherein secure computational vaults collect event logs devoid of upstream alterations and protect them at runtime, thus generating results derived directly from the original information source. Elkoumy et al. [83, 82] present Shareprom. Like our work, their solution offers a means for independent entities to execute process mining algorithms in inter-organizational settings while safeguarding the proprietary input data from exposure to external parties operating within the same context. Shareprom's functionality, though, is confined to the execution of operations involving event log abstractions [8] represented as directed acyclic graphs, which the parties employ as intermediate pre-elaboration to be fed into secure multiparty computation (SMPC) [53]. As the authors remark, relying on this specific graph representation imposes constraints that may turn out to be limiting in a number of process mining scenarios. In contrast, our approach allows for the secure, ciphered transmission of event logs (or segments thereof) to process mining nodes. Moreover, SMPC-based solutions require computationally intensive operations and synchronous cooperation among multiple parties, which make these protocols challenging to manage as the number of participants scales up [180]. In our research work, individual computing nodes run the calculations, thus not requiring synchronization with other machines once the input data is loaded.

We are confronted with the imperative task of integrating event logs originating from different data sources and reconstructing consistent traces that describe collaborative process executions. Consequently, we engage in an examination of methodologies delineated within the literature, each of which offers insights into the merging of event logs within inter-organizational settings. The work of Claes et al. [49] holds particular significance for our research efforts. Their seminal study introduces a two-step mechanism operating at the structured data level, contingent upon the configuration and subsequent application of merging rules. Each such rule indicates the relations between attributes of the traces and/or the activities that must hold across distinct traces to be combined. In accordance with their principles, our research incorporates a structured data-level merge based on case references and timestamps as merging attributes. The research by Hernandez et al. [100] posits a methodology functioning at the raw data level. Their approach represents traces and activities as *bag-of-words* vectors, subject to cosine similarity measurements to discern links and relationships between the traces earmarked for combination. An appealing aspect of this approach lies in its capacity to generalize the challenge

of merging without necessitating a-priori knowledge of the underlying semantics inherent to the logs under consideration. However, it entails computational overhead in the treatment of data that can interfere with the overall effectiveness of our approach.

5.3 Confidential Computing and Trusted Execution Environments

Confidential computing encompasses a suite of techniques aimed at safeguarding data while it is in use. These methods leverage hardware-backed protection of highly sensitive computations within attested *Trusted Execution Environments* (TEEs). According to the Confidential Computing Consortium (CCC) [1], TEEs are secure execution contexts that provide assurance regarding three key properties: (*data confidentiality*) unauthorized entities cannot view data while it is in use within the TEE, (*data integrity*) unauthorized entities cannot add, remove, or alter data while it is in use within the TEE, and (*code integrity*) unauthorized entities cannot add, remove, or alter code executing in the TEE [159].

TEE technologies are diverse, and their features may differ depending on the specific implementation of the underlying Central Processing Unit (CPU).

TEEs differ primarily in the extension of their trusted surface, that is, the hardware-enforced boundary defining the extent of isolation. Two main classes can be distinguished based on this criterion [108]. The first includes application-level (or process-level) TEEs, which protect a restricted portion of an application's address space. In such architectures, implemented among others by Intel SGX[2] and ARM TrustZone[3], only specific regions of memory explicitly designated as trusted are protected with confidentiality and integrity guarantees, whereas the remaining system components are treated as untrusted [21]. The second class comprises virtual-machine level TEEs, such as AMD SEV[4] and Intel TDX,[5] which extend the isolation boundary to an entire guest virtual machine. These TEEs protect memory and CPU state belonging to the virtual machine from inspection or tampering by (even higher-privileged) external software [135].

Another distinguishing characteristic among TEE technologies lies in whether the memory regions allocated to the isolated environment are protected through hardware-based encryption. In TEE technologies such as Intel SGX, AMD SEV, and

[1] confidentialcomputing.io. Accessed: May 24, 2026
[2] software-guard-extensions/overview. Accessed: May 24, 2026
[3] arm.com/technologies/trustzone. Accessed: May 24, 2026
[4] amd.com/sev. Accessed: May 24, 2026
[5] intel.com/trust-domain-extensions/overview. Accessed: May 24, 2026

Intel TDX, the CPU encrypts and protects the integrity of data within the trusted domain. This feature relies on dedicated sections of a CPU capable of handling encrypted data within a reserved section of the main memory [52]. This protected memory section is encrypted by the CPU using a random key derived at every power cycle. In contrast, technologies like ARM TrustZone mainly rely on access control layers between trusted and untrusted execution modes, with no hardware-based encryption involved.

A further essential feature characterizing certain TEE technologies, including Intel SGX, Intel TDX, and AMD SEV, is *attestability*. Through this feature, the CPU can produce a cryptographic proof, typically referred to as *attestation evidence*, to demonstrate the trusted status of the TEE to a local or remote verifier. Following the Remote ATtestation procedureS (RATS) standard [34], remote attestation protocols generally consist of the following steps: (1) The *verifier* establishes a communication channel with the *attester* (which may correspond to a trusted application or a virtual machine) and requests the generation of the attestation evidence. (2) The attester then instructs the CPU to produce the attestation evidence, which serves as cryptographic proof of its trusted state. In most protocols, the attestation evidence takes the form of a digital signature derived from hardware-bound secrets and includes a *measurement*, namely a cryptographic hash representing the state of the virtual machine or the trusted application's code. The attestation evidence is signed using a hardware-protected attestation key embedded in the processor. Depending on the TEE technology, the verifier may rely on the CPU manufacturer to validate this signature using the corresponding public endorsement credentials and to confirm the integrity of the reported measurement. (3) Upon receiving the attestation evidence, the verifier validates its authenticity by relying on an *endorser* (typically the TEE manufacturer) responsible for providing the necessary public credentials and reference values. The verifier subsequently compares the reported measurement against one or more reference values to reach a trust decision. The successful completion of this process attests to the integrity and authenticity of the entity providing the attestation evidence.

Attestable TEE implementations enable trusted virtual machines and applications to embed custom data into the attestation evidence at the time of its generation (see step 2 above). This feature allows verifiers to apply tailored appraisal policies by evaluating the included information. Furthermore, this custom data field can serve to authenticate the trusted origin of cryptographic material, which may subsequently be leveraged to establish secure communication channels once the attestation evidence has been successfully verified [99]. While CPU manufacturers such as Intel and AMD provide web services supplying the validation material required to verify

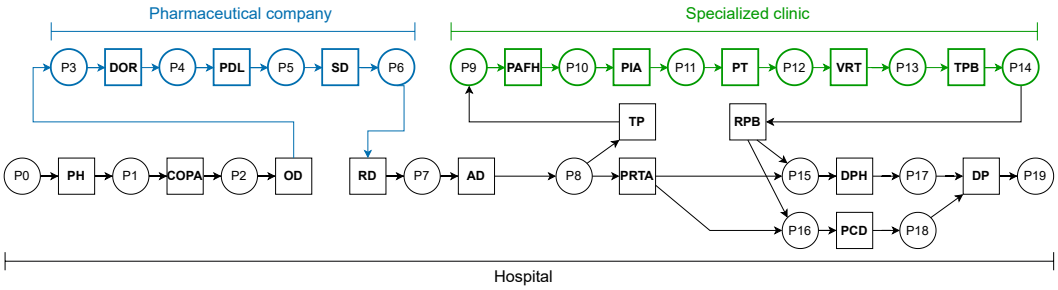


Figure 5.1. Collaborative Workflow net of a simplified healthcare scenario

Table 5.1. Events from cases 312 (Alice) and 711 (Bob) recorded by the Hospital, the Specialized clinic, and the Pharmaceutical company

Hospital

	Case	Timestamp	Activity
e_1	312	2022-07-14T10:36	PH
e_2	312	2022-07-14T16:36	COPA
e_3	711	2022-07-14T17:21	PH
e_4	312	2022-07-14T17:36	OD
e_5	711	2022-07-14T23:21	COPA
e_6	711	2022-07-15T00:21	OD
e_7	711	2022-07-15T18:55	RD
e_8	312	2022-07-15T19:06	RD
e_9	711	2022-07-15T20:55	AD
e_{10}	312	2022-07-15T21:06	AD

	Case	Timestamp	Activity
e_{11}	312	2022-07-15T22:06	TP
e_{12}	711	2022-07-16T00:55	PRTA
e_{13}	711	2022-07-16T01:55	PCD
e_{14}	711	2022-07-16T02:55	DPH
e_{15}	711	2022-07-16T04:55	DP
e_{16}	312	2022-07-16T07:06	RPB
e_{17}	312	2022-07-16T09:06	DPH
e_{18}	312	2022-07-16T10:06	PCD
e_{19}	312	2022-07-16T11:06	DP

Pharmaceutical company

	HospitalCaseID	Timestamp	Activity
e_{20}	312	2022-07-15T09:06	DOR
e_{21}	711	2022-07-15T09:30	DOR
e_{22}	312	2022-07-15T11:06	PDL
e_{23}	711	2022-07-15T11:30	PDL
e_{24}	312	2022-07-15T13:06	SD
e_{25}	711	2022-07-15T13:30	SD

Specialized clinic

	TreatmentID	Timestamp	Activity
e_{26}	312	2022-07-16T00:06	PAFH
e_{27}	312	2022-07-16T01:06	PIA
e_{28}	312	2022-07-16T03:06	PT
e_{29}	312	2022-07-16T04:06	VRT
e_{30}	312	2022-07-16T05:06	TPB

T_{312} =⟨ PH, COPA, OD, DOR, PDL, SD, RD, AD, TP, PAFH, PIA, PT, VRT, TPB, RPB, DPH, PCD, DP⟩
T_{711} =⟨ PH, COPA, OD, DOR, PDL, SD, RD, AD, PRTA, PCD, DPH, DP⟩

attestation evidence generated by their TEE-enabled processors, they delegate to system designers the responsibility of defining appropriate trust-decision policies and establishing secure communication channels [136].

In this work, we consider application-level TEEs that provide hardware-backed memory encryption and support remote attestation. Accordingly, our contribution implicitly assumes TEEs endowed with these three essential characteristics.

5.4 Collaborative Scenario

To provide a running example and motivating scenario for our investigation, we focus on a simplified hospitalization process for the treatment of rare diseases. The process model is depicted as a Workflow net in Fig. 5.1 and involves the cooperation of three parties: a Hospital, a Pharmaceutical company, and a Specialized clinic. For the sake of simplicity, we describe the process through two cases, recorded by the information systems as in Sect. 5.4. Alice's journey (**case 312**) begins when she enters the hospital for the preliminary examinations (patient hospitalized, PH). The

Table 5.2. DECLARE specification generated from the Hospital $\mathcal{WN}$

AtMostOne (tPH)	End (tDP)	Alt.Prec. (tOD, tRD)
Alt.Prec. (tTP, tRPB)	Alt.Prec. (tCPA, tOD)	Alt.Prec. (tAD, {tPRTA, tTP})
Alt.Prec. (tDPH, tDP)	Alt.Prec. (tPCD, tDP)	Alt.Prec. ({tRPB, tPRTA}, tPCD)
Alt.Prec. (tPH, tCPA)	Alt.Prec. (tRD, tAD)	Alt.Prec. ({tRPB, tPRTA}, tDPH)

Table 5.3. DECLARE specification generated from the Pharmaceutical company $\mathcal{WN}$

End (tSD)	AtMostOne (tDOR)	Alt.Prec. (tDOR, tPDL)	Alt.Prec. (tPDL, tSD)

Hospital then places an order for the drugs (OD) to the Pharmaceutical company for treating Alice's specific condition. Afterwards, the Pharmaceutical company acknowledges that the drug order is received (DOR), proceeds to produce the drugs in the laboratory (PDL), and ships the drugs (SD) back to the Hospital. Upon receiving the medications, the Hospital administers the drug (AD), and conducts an assessment to determine if Alice can be treated internally. If specialized care is required, the Hospital transfers the patient (TP) to the Specialized clinic. When the patient arrives from the hospital (PAFH), the Specialized clinic performs in-depth analyses (PIA) and proceeds with the treatment (PT). Once the Specialized clinic has completed the evaluations and verified the response to the treatment (VRT), it transfers the patient back (TPB). The Hospital receives the patient back (RPB) and prepares the clinical documentation (PCD). If Alice has successfully recovered, the Hospital declares the patient as healed (DPH). When Alice's treatment is complete, the Hospital discharges the patient (DP). Bob (**case 711**) enters the Hospital a few hours later. His hospitalization process is similar to Alice's. However, he does not need specialized care, and his case is only treated by the Hospital. Therefore, the Hospital performs the response to treatment analyses (PRTA) instead of transferring him to the Specialized clinic.

Both the National Institute of Statistics of the country in which the three organizations reside and the University that hosts the hospital wish to uncover information on this inter-organizational process for reporting and auditing purposes via process analytics [107]. The involved organizations share the urge for such an analysis and wish to be able to repeat the analytic task also in-house. The Hospital, the Specialized clinic, and the Pharmaceutical company have a partial

Table 5.4. DECLARE specification generated from the Specialized clinic $\mathcal{WN}$

End (tTPB)	AtMostOne (tPAFH)	Alt.Prec. (tPT, tVRT)	Alt.Prec. (tVRT, tTPB)
Alt.Prec. (tPAFH, tPIA)	Alt.Prec. (tPIA, tPT)		

Table 5.5. DECLARE specification generated from the whole $\mathcal{WN}$

ATMOSTONE (tPH)	END (tDP)	ALT.PREC. (tPH, tCOPA)
ALT.PREC. (tCOPA, tOD)	ALT.PREC. (tOD, tDOR)	ALT.PREC. (tDOR, tPDL)
ALT.PREC. (tPDL, tSD)	ALT.PREC. (tSD, tRD)	ALT.PREC. (tRD, tAD)
ALT.PREC. (tAD, {tPRTA, tTP})	ALT.PREC. (tTP, tPAFH)	ALT.PREC. (tPAFH, tPIA)
ALT.PREC. (tPIA, tPT)	ALT.PREC. (tPT, tVRT)	ALT.PREC. (tVRT, tTPB)
ALT.PREC. (tTPB, tRPB)	ALT.PREC. (tDPH, tDP)	ALT.PREC. ({tPRTA, tRPB}, tDPH)
ALT.PREC. ({tPRTA, tRPB}, tPCD)	ALT.PREC. (tPCD, tDP)	

view of the overall unfolding of the inter-organizational process as they record the events stemming from the parts of their pertinence. In Sect. 5.4, we show cases 312 and 711 and the corresponding traces recorded by the Hospital (i.e., T^H_{312} and T^H_{711}), the Specialized clinic (i.e., T^S_{312} and T^S_{711}), and the Pharmaceutical company (i.e., T^C_{312} and T^C_{711}). Those traces are projections of the two combined ones for the whole inter-organizational process: T_{312} =⟨PH, COPA, OD, DOR, PDL, SD, RD, AD, TP, PAFH, PIA, PT, VRT, TPB, RPB, DPH, PCD, DP⟩ and T_{711} =⟨PH, COPA, OD, DOR, PDL, SD, RD, AD, PRTA, PCD, DPH, DP⟩. Each organization can therefore represent and verify the behavior through a local declarative specification, while remaining interoperable with the other parties when a joint analysis is permitted. To illustrate this modular viewpoint, Tabs. 5.2 to 5.4 report the specifications generated from the Workflow nets of the Hospital, the Pharmaceutical company, and the Specialized clinic, respectively. While Tab. 5.5 depicts the joint whole process specification.

Results stemming from the analysis of the local cases would not provide a full picture. Assessing conformance to the overall process requires composing these local constraints and reasoning on the end-to-end behavior. Following the findings drawn in Chapter 4, interactions among rules may only emerge at the global specification level. Thus, data should be merged.

However, to safeguard the confidentiality of the information, the involved parties cannot give other organizations open access to their traces. The diverging interests (being able to run process mining and conformance checking algorithms on data from multiple sources without giving away the local event logs in clear) motivate our research. We remark that the problem we aim to solve spans across an array of domains beyond healthcare. It particularly applies to scenarios in which one or more parties are interested in process analytics outcomes based on data they bear but cannot be disclosed to the other process actors or to the miners. In the supply chain realm, e.g., the extraction of aggregate knowledge about trends and management guidelines is called for, but the acquisition of competitive advantage out of knowledge leakage must be prevented [167]. In personal informatics, company-wide work routine

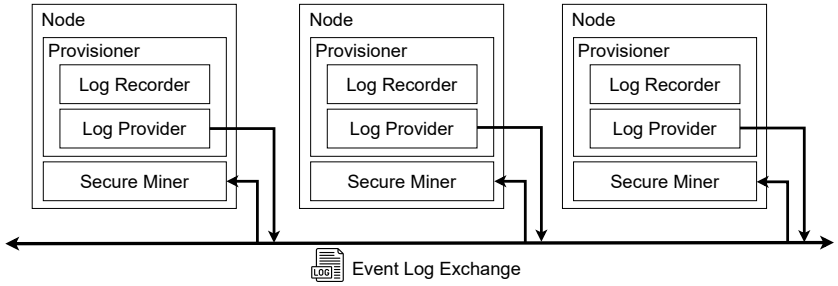


Figure 5.2. The CONFINE high-level architecture

monitoring and analysis are desirable, though the details of individual participants should be sheltered from inquisitive inspections [162].

5.5 Design

Our goal is to enable the secure aggregation and elaboration of original, unaltered event logs from decentralized sources in dedicated environments that potentially lie beyond the individual organizations' information perimeter. With this objective in mind, we devise the `Secure Miner` component, which is capable of safeguarding data merge and processing by running certified code in an isolated execution vault. Thus, we decouple provisioning from treatment, and the two tasks can be carried out by distinct computing nodes. Here, we introduce CONFINE's key components, with a special focus on the `Secure Miner`.

The CONFINE architecture at large. Our architecture involves different information systems running on multiple machines. An organization can take at least one of the following roles: **provisioning** if it delivers local event logs to be collaboratively mined; **mining** if it applies process mining algorithms using event logs retrieved from provisioners. Figure 5.2 depicts the high-level schematization of the CONFINE framework. In our solution, each organization hosts one or more nodes encompassing diverse components (the names of which will henceforth be formatted with a `teletype` font). Depending on the played role, nodes come endowed with a `Provisioner` or a `Secure Miner`, or both. The `Provisioner` component, in turn, consists of the following two sub-components. The `Log Recorder` registers the events taking place in the organizations' systems. The `Log Provider` delivers on-demand data to miners. The Hospital and all other parties in our example record Alice and Bob's cases using the `Log Recorder`. The `Log Recorder`, in turn, is queried by the `Log Provider` for event logs to be made available for mining. The latter controls access to local event logs by authenticating data requests by miners and rejecting those that come from unauthorized parties. In our motivating scenario,

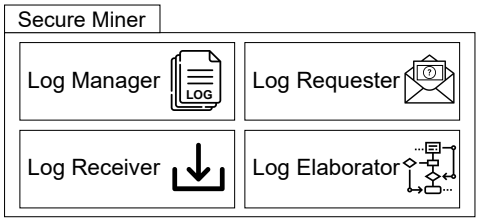


Figure 5.3. Sub-components of the Secure Miner

the Specialized clinic, the Pharmaceutical company, and the Hospital leverage `Log Providers` to authenticate the miner before sending their logs. The `Secure Miner` component shelters external event logs inside a protected environment to preserve data confidentiality and integrity. Notice that `Log Providers` accept requests issued solely by `Secure Miners`. Next, we provide an in-depth focus on the latter.

The Secure Miner. The primary objective of the `Secure Miner` is to allow miners to securely execute process mining algorithms using event logs retrieved from provisioners (the Specialized clinic, the Pharmaceutical company, and the Hospital in our example). `Secure Miners` are isolated components that guarantee data inalterability and confidentiality.

Figure 5.3 illustrates a schematization of the `Secure Miner`, which consists of four sub-components: *(i)* the `Log Requester`; *(ii)* the `Log Receiver`; *(iii)* the `Log Manager`; *(iv)* the `Log Elaborator`. The `Log Requester` and the `Log Receiver` are the sub-components that we employ during the event log retrieval. `Log Requesters` send authenticable data requests to the `Log Providers`. The `Log Receiver` collects event logs sent by `Log Providers` and entrusts them to the `Log Manager`, securing them from accesses that are external to the `Secure Miner`. Miners of our motivating scenario, such as the University and the National Institute of Statistics, employ these three components to retrieve and store Alice and Bob's data. The `Log Manager` merges the event data locked in the `Secure Miner` to have a global view of the inter-organizational process comprehensive of activities executed by each involved party. The `Log Elaborator` executes process mining algorithms in a protected environment, inaccessible from the outside computation environment. In our motivating scenario, the `Log Manager` combines the traces associated with the cases of Alice (i.e., T^H_{312}, T^S_{312}, and T^C_{312}) and Bob (i.e, T^H_{711}, T^S_{711}, and T^C_{711}), generates the chronologically sorted traces T_{312} and T_{711}, and feeds them into the `Log Elaborator`'s mining algorithms (see the bottom-right quadrant of Sect. 5.4).

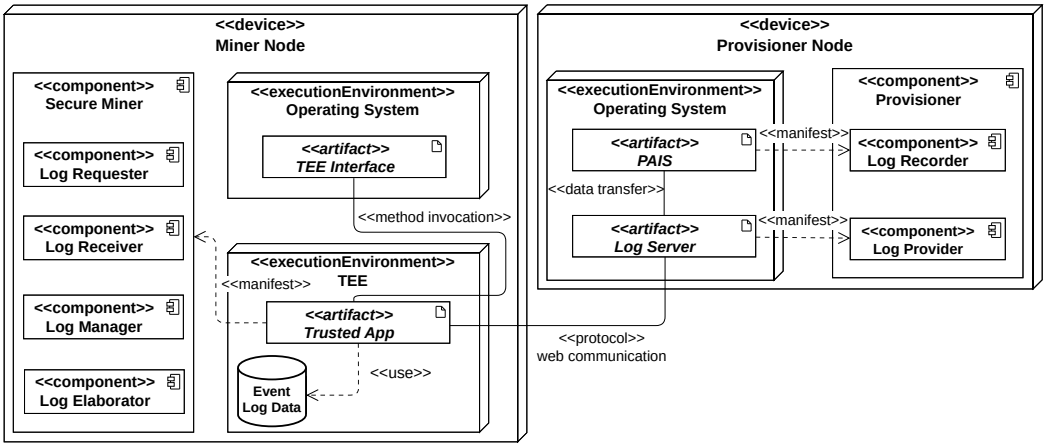


Figure 5.4. UML deployment diagram of the CONFINE architecture

5.6 Realization

Thus far, we have outlined the main functionalities of each component at large. Here we discuss the technical aspects concerning the realization of our solution. We first present the technologies through which we enable the design principles in Sect. 5.5. Then, we discuss the CONFINE interaction protocol. Finally, we show the implementation details.

5.6.1 Deployment

Figure 5.4 depicts a UML deployment diagram [111] to illustrate the employed technologies and computation environments. We recall that the `Miner` and `Provisioner` nodes are drawn as separated, although organizations can host both. In our motivating scenario, e.g., the Hospital can be equipped with machines aimed for both mining and provisioning.

`Provisioner Node`s host the `Provisioner`'s components, i.e., the `Log Recorder` and the `Log Provider`. The Process-Aware Information System (`PAIS`) manifests the `Log Recorder` [79]. The `PAIS` grants access to the `Log Server`, enabling it to retrieve event log data. The `Log Server`, on the other hand, embodies the functionalities of the `Log Provider`, implementing services that handle remote data requests and provide event log data to the miners. The `Miner Node` is characterized by two distinct *execution environments*: the `Operating System` (`OS`) and the Trusted Execution Environment (`TEE`) [159]. `TEE`s establish isolated contexts separate from the `OS`, safeguarding code and data through hardware-based encryption mechanisms. This technology relies on dedicated sections of a CPU that handle encrypted data within a reserved section of the main memory [52]. By enforcing memory access restrictions, TEEs aim to prevent one application from reading or altering the

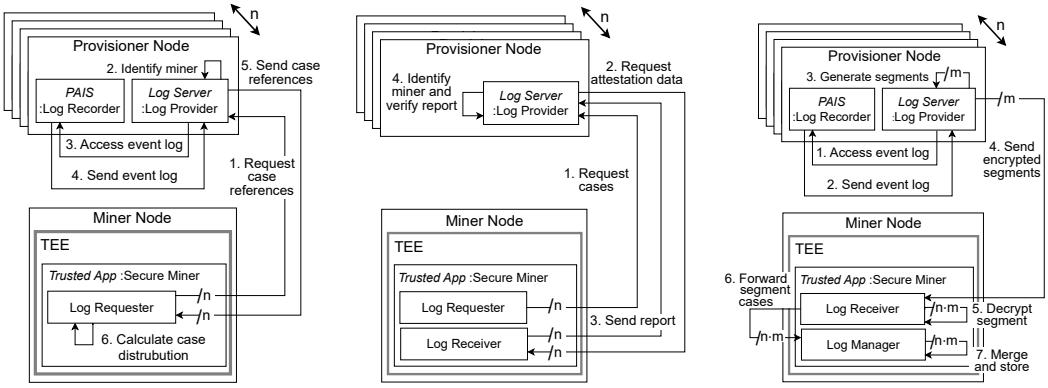
memory space of another, thus enhancing system security. These dedicated areas in memory are limited, though. Once the limits are exceeded, TEEs have to scout around in outer memory areas, thus conceding the opportunity to malicious readers to understand the saved data based on the reads and writes. To avoid this risk, TEE implementations often raise errors that halt the program execution when memory demand goes beyond the available space. Therefore, the design of secure systems that resort to TEEs must take into account that memory consumption must be kept under control.

We leverage the security guarantees provided by TEEs [108] to protect a `Trusted App` responsible for fulfilling the functions of the `Secure Miner` and its associated sub-components. Our `TEE` component ensures the integrity of the `Trusted App` code, protecting it against potential malicious manipulations and unauthorized access by programs running within the `OS`. Additionally, we utilize the isolated environment of the `TEE` to securely store event log data (e.g., Alice and Bob's cases). The `TEE` retains a private key in its inaccessible memory section, paired with a public key in a Rivest-Shamir-Adleman (RSA) [154] scheme for attestation (only the owner of the private key can sign messages in a way that is verifiable via the public key) and secure message encryption (only the owner of the private key can decode messages that are encrypted with the corresponding public key). In our solution, access to data located in the `TEE` is restricted to the sole `Trusted App`. Users interact with the `Trusted App` through the `TEE Interface`, which serves as the exclusive communication channel. The `Trusted App` offers secure methods, invoked by the `Trusted App Interface`, for safely receiving information from the `OS` and outsourcing the results of computations.

5.6.2 The CONFINE protocol

We orchestrate the interaction of the components in CONFINE via a protocol, which consists of four subsequent stages: *(i) initialization*, *(ii) remote attestation*, *(iii) data transmission*, and *(iv) computation*. These stages are depicted in Figs. 5.5(a) to 5.5(c) and 5.6, respectively. They are mainly enacted by a `Miner Node` (multiple instances of which can be deployed in a decentralized fashion) and n `Provisioner Nodes`. We assume their communication channel is reliable [40] and secure [110]. In the following, we describe each of the above phases in detail.

Initialization. The objective of the initialization stage is to inform the miner about the distribution of cases related to a business process among the `Provisioner Nodes`. At the onset of this stage, the `Log Requester` within the `Trusted App` issues n requests, one per `Log Server` component, to retrieve the list of case references they record (step 1 in Fig. 5.5(a)). Following sender authentication (2), each `Log`



(a) Initialization **(b)** Remote attestation **(c)** Data transmission

Figure 5.5. Unfolding example for the initialization, remote attestation and data transmission phases of the CONFINE protocol

`Server` retrieves the local event log from the `PAIS` (3, 4) and subsequently responds to the `Log Requester` by providing a list of its associated case references (5). After collecting these n responses, the `Log Requester` delineates the distribution of cases. In the context of our motivating scenario, by the conclusion of the initialization, the miner gains knowledge that the case associated with Bob, synthesized in the traces T^H_{711} and T^C_{711}, is exclusively retained by the Hospital and the Specialized Clinic. In contrast, the traces of Alice's case, denoted as T^H_{312}, T^C_{312}, and T^S_{312}, are scattered across all three organizations.

Remote attestation. The remote attestation serves the purpose of establishing trust between miners and provisioners in the context of fulfilling data requests. This phase adheres to the overarching principles outlined in the RATS RFC standard [35] serving as the foundation for several TEE attestation schemes (e.g., Intel EPID[6] and AMD SEV-SNP[7]). Remote attestation has a dual objective: *(i)* to furnish provisioners with compelling evidence that the data request for an event log originates from a `Trusted App` running within a TEE; *(ii)* to confirm the specific nature of the `Trusted App` as an authentic `Secure Miner` software entity. This phase is triggered when the `Log Requester` sends a new case request to the `Log Server`, specifying: *(i)* the segment size (henceforth, *seg_size*), and *(ii)* the set of the requested case references. Both parameters will be used in the subsequent *data transmission* phase. Each of the n `Log Servers` commences the verification process by requesting the necessary information from the `Log Receiver` to conduct the attestation (2). Subsequently, the `Log Receiver` generates the attestation report containing the so-called *measurement* of the `Trusted App`, which is defined as the hash value of

[6] sgx101.gitbook.io/sgx101/sgx-bootstrap/attestation. Accessed: May 24, 2026.
[7] amd.com/en/processors/amd-secure-encrypted-virtualization. Accessed: May 24, 2026.

the combination of its source code and data. Once this report is signed using the attestation private key associated with the TEE's hardware of the Miner Node, it is transmitted by the Log Receiver to the Log Servers alongside the attestation public key of the Miner Node (3). The Log Servers authenticate the miner using the public key and decrypt the report (4). In this last step, the Log Servers undertake a comparison procedure in which they juxtapose the measurement found within the decrypted report against a predefined reference value associated with the source code of the Secure Miner. If the decrypted measurement matches the predefined value, the Miner Node gains trust from the provisioner.

Data transmission. Once the trusted nature of the Trusted App is verified, the Log Servers proceed with the transmission of their cases. To accomplish this, each Log Server retrieves the event log from the PAIS (steps 1 and 2, in Fig. 5.5(c)), and filters it according to the case reference set specified by the miner. Given the constrained workload capacity of the TEE, Log Servers could be requested to partition the filtered event log into m distinct segments. Log segments contain a variable count of entire cases (3). The cumulative size of these segments is governed by the threshold parameter specified by the miner in the initial request (step 1 of the remote attestation phase, Fig. 5.5(b)). As an illustrative example from our motivating scenario, the Log Server of the Hospital may structure the segmentation such that T^H_{312} and T^H_{711} are in the same segment, whereas the Specialized clinic might have T^S_{312} and T^S_{711} in separate segments. Subsequently, the n Log Servers transmit their m encrypted segments to the Log Receiver of the Trusted App (4). The Log Receiver, in turn, collects the $n \times m$ responses in a queue, processing them one at a time. After decrypting a processed segment (5), the Log Receiver forwards the cases contained therein to the Log Manager (6). Data belonging to the same process instance are merged by the Log Manager to build a single trace (e.g., T_{312}) comprehensive of all the events in the partial traces (T^H_{312}, T^S_{312} and T^C_{312}). To do so, the Log Manager applies a specific *merging schema* (i.e., a rule specifying the attributes that identify a case) as stated in [49]. In our illustrative scenario, the merging schema to combine the cases of Alice is contingent upon the linkage established through their case identifier (312). We underline that our solution facilitates the incorporation of diverse merging schemas encompassing distinct trace attributes. The outcomes arising from merging the cases within the processed segments are securely stored by the Log Manager in the TEE.

Computation. The Trusted App requires all the provisioners to have delivered data referring to the same process instances. For example, when the Hospital and the other organizations have all delivered their information concerning case 312 to the Trusted App, the process instance associated with Alice becomes eligible for

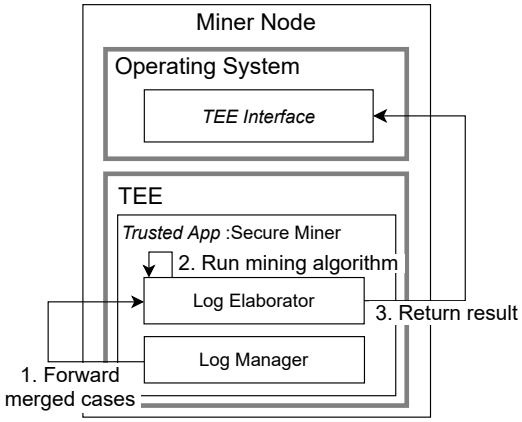


Figure 5.6. Computation phase of the CONFINE protocol

the computation phase, illustrated in Fig. 5.6. The `Log Manager` forwards the cases earmarked for computation to the `Log Elaborator` (step 1).

These cases may constitute either the entire merged event log or a subset thereof. The former setting entails a single computation routine, thus saving execution time but requiring a larger memory buffer in the `TEE`, whereas the latter necessitates multiple consecutive elaborations with a lower demand for space.

Subsequently, the `Log Elaborator` proceeds to input the merged cases into the process mining algorithm (2).

Notice that the above choice on the buffering of cases affects the selection of the mining algorithm to employ. If we elaborate subsequent batches, each containing a part of all merged cases, the mining algorithm must support incremental processing, enriching the output as new batches come along. An example of this class of algorithms is the HeuristicsMiner [173]. Otherwise, incrementality is not required.

Ultimately, the outcome of the computation is relayed by the `Log Elaborator` from the `TEE` to the `TEE Interface` running atop the `Operating System` of the `Miner Node` (3). In our motivating scenario, the University and the National Institute of Statistics, serving as miners, disseminate the outcomes of computations, generating analyses that benefit the provisioners, although the original data are never revealed in clear. Furthermore, our protocol enables the potential for provisioners to have their own `Secure Miner`, allowing them autonomous control over the computed results. Notice that the CONFINE protocol does not impose restrictions on the post-computational handling of results.

Table 5.6. Event logs used for our experiments

Name	Type	Activities	Cases	Max events	Min events	Avg. events	Organization ↦ Activities
Motivating scenario	Synthetic	19	1000	18	9	14	$\mathscr{O}^P \mapsto 3$, $\mathscr{O}^C \mapsto 5$, $\mathscr{O}^H \mapsto 14$
Sepsis [134]	Real	16	1050	185	3	15	$\mathscr{O}^1 \mapsto 1$, $\mathscr{O}^2 \mapsto 1$, $\mathscr{O}^3 \mapsto 14$
BPIC 2013 [164]	Real	7	1487	123	1	9	$\mathscr{O}^1 \mapsto 6$, $\mathscr{O}^2 \mapsto 7$, $\mathscr{O}^3 \mapsto 6$

5.7 Evaluation

In this section, we report on our evaluation of CONFINE. First, we introduce our implementation in Sect. 5.7.1. Then, we propose an experimental verification of Sect. 5.7.2, empirically demonstrating the output convergence. Finally, we evaluate our implementation with runtime tests on memory usage to show that our approach determines low memory demand, compatible with the constrained capacity of TEEs (Sect. 5.7.3).

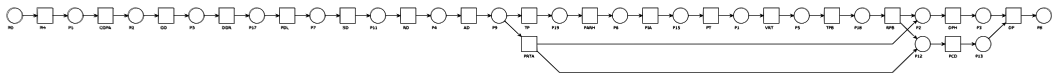
5.7.1 Implementation

We implemented the `Secure Miner` component as an Intel SGX [25] trusted application, encoded in Go through the EGo framework.[8] We opted for an Intel SGX TEE for our prototype because its features align with the TEE profile outlined in Sect. 5.3 and offers a well-documented developer ecosystem. These characteristics made Intel SGX a practical choice for implementing our research prototype aimed at demonstrating the feasibility of CONFINE. However, we highlight that recent studies have reported vulnerabilities [141, 37], and real-world deployments may therefore prefer more robust alternatives or incorporate tailored mitigation strategies to address these concerns [88].

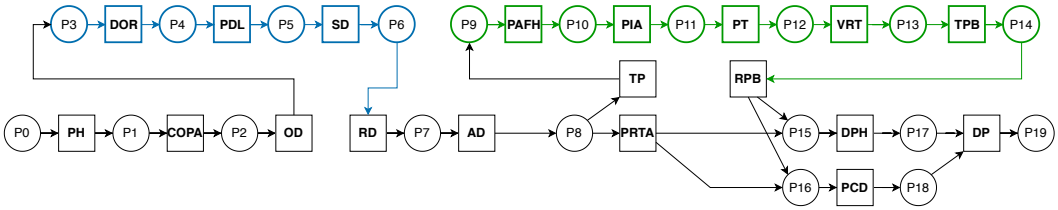
We encoded the provisioner's `Log Server` in GO. The implementation of our `Secure Miner` component facilitates input reception from users via a console application (i.e., the `TEE Interface` in Sect. 5.6.1). We store the descriptive information of the `Trusted App` (i.e., heap size, embedded files, and environment variables) in a JavaScript Object Notation (JSON) file. In our implementation, the `Secure Miner` app maintains references to provisioners' `Log Servers` in a dedicated JSON file, alongside the label of the case ids attribute in their respective log partitions. We embed in the `TEE` during deployment. We resort to a Transport Layer Security (TLS) [169] communication channel between miners and provisioners over the HyperText Transfer Protocol (HTTP) web protocol to secure the information exchange and authenticate the miner's organization.

We integrate two established process mining algorithms within our `Secure Miner`

[8] `https://docs.edgeless.systems/ego`. Accessed: December 2, 2025.



(a) Raw model as returned by the miner.



(b) Refined rendering for readability.

Figure 5.7. *HeuristicsMiner* output in CONFINE

implementation: *HeuristicsMiner* [173] and *DeclareConformance* [73] from the *pm4py* library.[9] The selected algorithms are representative of process mining techniques along two complementary dimensions. First, *HeuristicsMiner* and *DeclareConformance* perform the core process mining tasks of process discovery and conformance checking, respectively. Second, they reflect the two predominant modeling paradigms in process mining, namely *imperative* (as in the case of *HeuristicsMiner*) and *declarative* approaches (as in the case of *DeclareConformance*) [147, 72]. Through the integration of these algorithms, we demonstrate that our implementation is applicable to distinct process mining tasks and accommodates different methodological paradigms.

Upon receiving each complete case, our *HeuristicsMiner* implementation produces a causal net, which is subsequently transformed into the corresponding workflow net of the analyzed process. The `TEE` outputs the resulting models in the Petri Net Markup Language (PNML) format, a standard for representing workflow nets.[10] Similarly, our *DeclareConformance* implementation computes fitness scores, which are returned from the `TEE` in the form of JSON files. For both algorithms, we provide the following two versions. **Incremental:** In this version, the algorithm takes batches of the complete cases as an input and updates a partial result while data transmission is still ongoing; **Non-incremental:** The algorithm is triggered at the end of the data transmission, once all the cases are collected. Our implementation of CONFINE, including the *HeuristicsMiner* and the *DeclareConformance* in Go, is openly accessible at the following URL: `github.com/Process-in-Chains/CONFINE/`.

[9] `github.com/process-intelligence-solutions/pm4py`. Accessed: May 24, 2026
[10] `www.pnml.org`. Accessed: May 24, 2026

5.7.2 Convergence

To experimentally validate the output convergence, we run a *convergence* test. We created a synthetic event log consisting of 1000 cases of 14 events on average (see Tab. 5.6) by simulating the inter-organizational process[11] and we partitioned it into three sub-logs (one per involved organization), an excerpt of which is listed in Sect. 5.4. We run the stand-alone *HeuristicsMiner* on the former and process the latter through our CONFINE toolchain. As expected, the results converge and are depicted in Fig. 5.7(a) in the form of a workflow net [3]. For the sake of clarity, in Fig. 5.7(b) we color activities recorded by the organizations following the scheme of Tab. 5.6 (black for the Hospital, blue for the Pharmaceutical company, and green for the Specialized clinic).

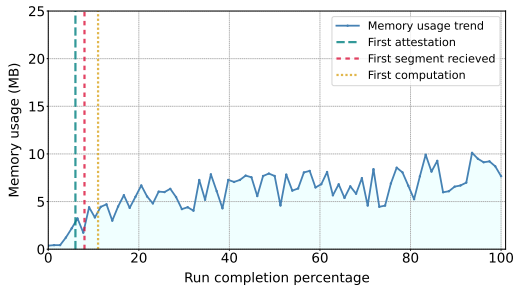
5.7.3 Performance Assessment

In what follows, we report on our experiments conducted to put our tool implementation to the test. As discussed in Sect. 5.6.1, the availability of space in the dedicated TEE areas is subject to hardware limitations. Therefore, we focus on memory consumption since exceeding those limits could diminish the level of security guaranteed by TEEs. We gauge the memory usage with synthetic and real-life event logs, to observe the trend during the enactment of our protocol. In particular, we focus on runtime consumption (Sect. 5.7.3), the effect that the segment size hyper-parameter discussed in Sect. 5.6.1 has on efficiency (Sect. 5.7.3), and scalability with respect to the event log size and the number of provisioners (Sect. 5.7.3). The raw execution data alongside the scripts to plot the obtained results are openly accessible at `github.com/Process-in-Chains/CONFINE/tree/main/evaluation/`.

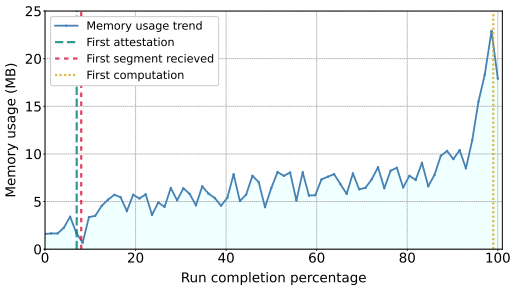
Runtime analysis

Figures 5.8(a) to 5.8(d) display the runtime memory utilization of our `Secure Miner` implementation (in MegaBytes) using the synthetic log derived from our motivating scenario as input. Figure 5.8(a) illustrates the memory utilization trend with the integration of the incremental variant of the HeuristicsMiner. Differently, Fig. 5.8(b) pictures the test outcome resulting from the execution of the protocol with the non-incremental variant of the algorithm. The dashed lines mark the starting points for the remote attestation, the data transmission, and the computation stages. We held the *seg_size* constant at 100 KiloBytes. In both figures, we observe that after the initiation of the data transmission stage (the dashed red line), the memory

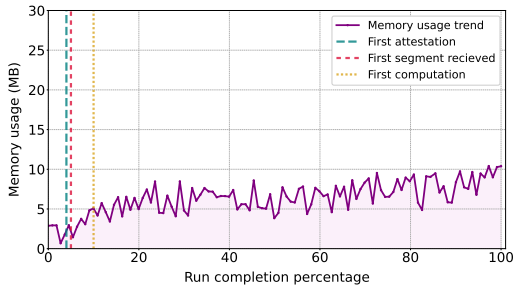
[11] We generated the event log through BIMP (`bimp.cs.ut.ee`; accessed: May 24, 2026). We filtered the log by retaining the sole events that report on the completion of activities, and removed the start and end events of the Pharmaceutical company and Specialized clinic's sub-processes.



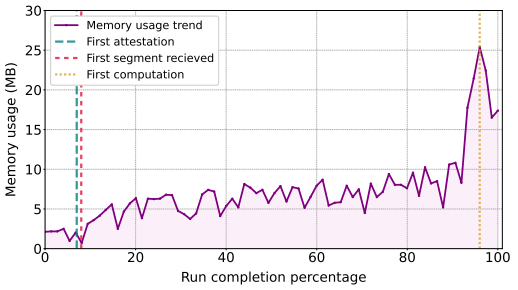
(a) Memory usage with incremental HeuristicsMiner



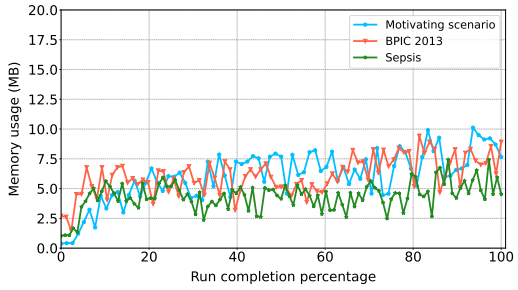
(b) Memory usage with the non-incremental HeuristicsMiner



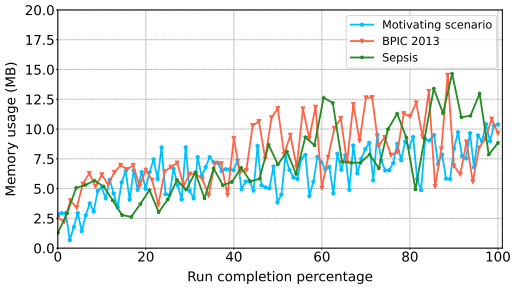
(c) Memory usage with incremental DeclareConformance



(d) Memory usage with non-incremental DeclareConformance

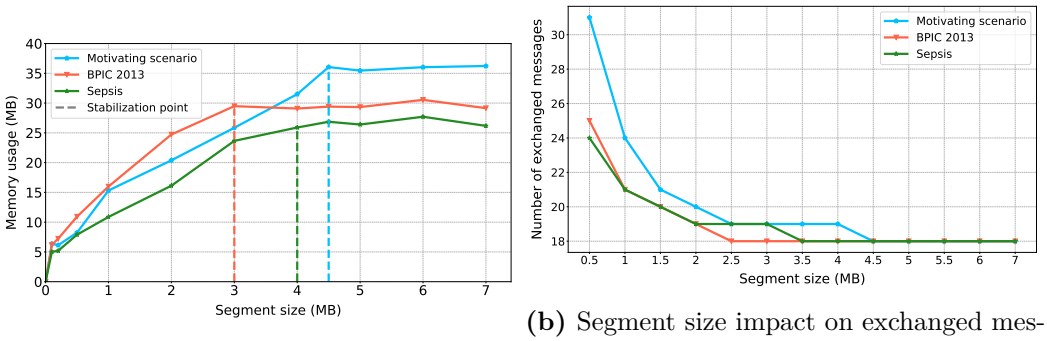


(e) Memory usage with three event logs and the incremental HeuristicsMiner



(f) Memory usage with three event logs and the incremental DeclareConformance

Figure 5.8. Memory usage test results



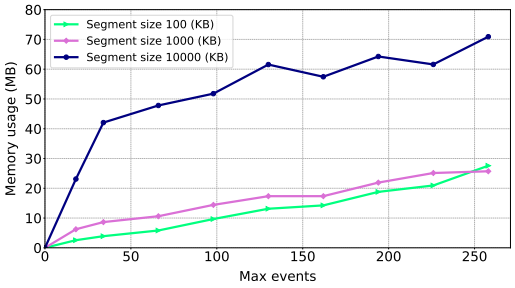
(a) Segment size impact on memory usage

(b) Segment size impact on exchanged messages overhead

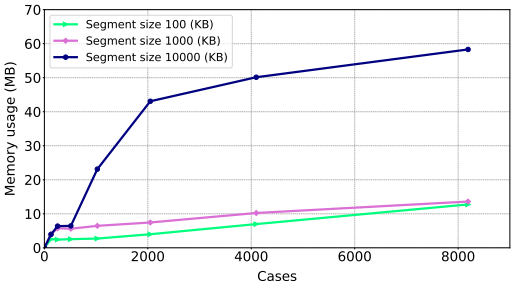
Figure 5.9. Segment size test result

utilization falls approximately between 5 and 10 MB. The major difference between the two trends concerns the computation phase. In Fig. 5.8(a), the HeuristicsMiner starts computing cases when the data transmission is still ongoing. This ensures the memory utilization trend remains within a constant range until the end of the run. In Fig. 5.8(b), the HeuristicsMiner is triggered after all the `Provisioners` have delivered their respective log partitions. Consequently, the final peak at the yellow dashed line signifies the computation of the entire event log, making the memory utilization diverge to a value of between 20 and 25 MB. This behavior is confirmed by Figs. 5.8(c) and 5.8(d), in which we consider the memory usage trend of the DeclareConformance algorithm variants. The tests indicate that the DeclareConformance demands a higher memory utilization than the HeuristicsMiner due to the involvement of the declarative process model in the conformance checking task.

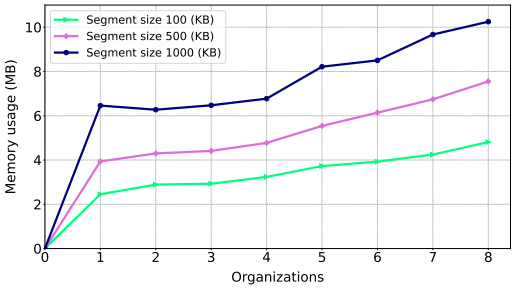
To verify whether these behaviors are due to the synthetic nature of our simulation-based event log, we also gauge the runtime memory usage of two public real-world event logs (Sepsis [134] and BPIC 2013 [164]). The characteristics of the event logs are summarized in Tab. 5.6. Since those are *intra-organizational* event logs, we split the contents to mimic an *inter-organizational* context. In particular, we separated the Sepsis event log based on the distinction between normal-care and intensive-care paths, as if they were conducted by two distinct organizations. Similarly, we processed the BPIC 2013 event log to sort it out into the three departments of the Volvo IT incident management system. Figures 5.8(e) and 5.8(f) depict the results with the integration of the incremental variants of HeuristicsMiner and DeclareConformance. We observe that the BPIC 2013 event log processing demands more memory probably owing to its larger size. Conversely, the Sepsis event log turns out to entail the least expensive run.



(a) Results of scalability test I



(b) Results of scalability test II



(c) Results of scalability test III

Test	**Seg.size**	$R^2_{\mathbf{lin}}$	$\widehat{\beta}$	$R^2_{\log}$
Max events	100	0.9847	0.0980	0.8291
	1000	0.9544	0.0821	0.9043
	10000	0.7357	0.1518	0.9386
Number of cases	100	0.9896	0.0013	0.6822
	1000	0.9629	0.0010	0.8682
	10000	0.7729	0.0068	0.9303
Provisioning organizations	100	0.9770	0.3184	0.8577
	500	0.9602	0.5174	0.7902
	1000	0.9066	0.6102	0.6977

(d) Scalability measurements

Figure 5.10. Scalability test results

Segment Size

To verify whether these trends are affected by the dimension of the exchanged data segments, we conducted additional tests to examine the trend of memory usage and the overhead of the exchanged message as the *seg_size* varies with all the aforementioned event logs. Notably, the polylines displayed in Fig. 5.9(a) indicate a linear increment of memory occupation until a breakpoint is reached. After that, the memory in use is steady. These points, marked by vertical dashed lines, correspond to the *seg_size* value that allows the provider's segments to be contained in a single data segment.

Figure 5.9(b) depicts the relationship between segment size and the number of message exchanges through CONFINE. As the segment size increases, there is a decrease in the number of messages exchanged, indicating a reduction in communication overhead. This trend is consistent across different logs and segment sizes, with a stabilizing effect observed after the segment size of 4.5 MB suggesting an optimal range to minimize message exchanges.

Scalability

In the remainder of this section, we examine the scalability of the `Secure Miner`, focusing on its capacity to efficiently manage an increasing workload in the presence of limited memory resources. We implemented three distinct test configurations gauging the average runtime memory usage as variations of our motivating scenario's log. In particular, we considered (I) the maximum number of events per case, (II) the number of cases ids, and (III) the number of provisioning organizations as independent integer variables. To conduct the test on the maximum number of events, we added a loop back from the final to the initial activity of the process model, progressively increasing the number of iterations $2 \leqslant x_{\circlearrowleft} \leqslant 16$ at a step of 2, resulting in $18 + 16 \cdot (x_{\circlearrowleft} - 1)$ events. Concerning the test on the number of cases, we simulated additional process instances so that $|\widehat{\mathrm{IID}}| = 2^{x_{iid}}$ having $x_{iid} \in \{7, 8, \ldots, 13\}$. Finally, for the assessment of the number of organizations, the test necessitated the distribution of the process model activities' into a variable number of pools, each representing a different organization ($|\widehat{\mathcal{O}}| \in \{1, 2, \ldots, 8\}$). We parameterized the above configurations with three segment sizes (in KiloBytes): *seg_size* $\in \{100, 1000, 10000\}$ for tests I and II, and *seg_size* $\in \{100, 500, 1000\}$ for test III (the range is reduced without loss of generality to compensate the partitioning of activities into multiple organizations). To facilitate a more rigorous interpretation of the output trends across varying *seg_size* configurations, we employ two well-known statistical measures. As a primary measure of goodness-of-fit, we employ the coefficient of determination R^2 [27], which assesses the degree to which the observed data adheres to the linear (R^2_{lin}) and logarithmic (R^2_{log}) regressions derived from curve fitting approximations. To further delve into the analysis of trends with a high R^2_{lin}, we consider the slope $\widehat{\beta}$ of the approximated linear regression [19].

Tab. 9(d) lists the measurements we obtained. We describe them to elucidate the observed patterns. Figure 5.10(a) depicts the results of test I, focusing on the increase of memory utilization when the number of events in the event logs grows. We observe that the memory usage for *seg_size* 100 and 1000 (depicted by green and lilac lines, respectively) are quite similar, whereas the setting with *seg_size* 10,000 (blue line) exhibits significantly higher memory usage. For the settings with *seg_size* 100 and 1000, R^2_{lin} approaches 1, signifying an almost perfect approximation of the linear relation, against lower R^2_{log} values. In these test settings, $\widehat{\beta}$ is very low yet higher than 0, thus indicating that memory usage is likely to continue increasing as the number of max events grows. The configuration with *seg_size* 10,000 yields a higher R^2_{log} value, thus suggesting a logarithmic trend, hence a greater likelihood of stabilizing memory usage growth rate as the number of maximum events increases. In Fig. 5.10(b), we present the results of test II, assessing the impact of the number of

cases on the memory consumption. As expected, the configurations with *seg_size* set to 100 and 1000 exhibit a trend of lower memory usage than settings with *seg_size* 10,000. The R^2_{lin} score of the trends with *seg_size* 100 and 1000 indicate a strong linear relationship between the dependent and independent variables compared to the trend with *seg_size* 10,000, which is better described by a logarithmic regression ($R^2_{\text{log}} = 0.9303$). For the latter, the R^2_{log} value is higher than the corresponding R^2_{lin} thus suggesting that the logarithmic approximation is better suited to describe the trend. Differently from test I, the $\widehat{\beta}$ score associated with the linear approximations of the trends with *seg_size* 100 and 1000 approaches 0, indicating that the growth rate of memory usage as the number of cases increases is negligible. In Fig. 5.10(c), we present the results of test III, on the relation between the number of organizations and the memory usage. The chart shows that memory usage trends increase as provisioning organizations increase for all three segment sizes. The R^2_{lin} values for the three *seg_size*s are very high, indicating a strong positive linear correlation. The test with *seg_size* 100 exhibits the slowest growth rate, as corroborated by the lowest $\widehat{\beta}$ result (0.3184). For the configuration with *seg_size* 500, the memory usage increases slightly faster ($\widehat{\beta} = 0.5174$). With *seg_size* 1000, the overall memory usage increases significantly faster than the previous configurations ($\widehat{\beta} = 0.6102$). We derive from these findings that the `Secure Miner` may encounter scalability issues when handling settings with a large number of provisioning organizations. Further investigation is warranted to determine the precise cause of this behavior and identify mitigation strategies.

5.8 Discussion

Following the above empirical examination of our approach implementation, in the remainder of this section, we discuss CONFINE by addressing the implications associated with its practical instantiation.

5.8.1 Practical Implications

The applications of CONFINE span across an array of domains, including and beyond healthcare, in which our running example in Sect. 5.4 is rooted. It particularly applies to scenarios in which one or more organizations are interested in process analytics outcomes based on data they hold, but cannot be disclosed to others or to the miners. In healthcare, CONFINE can aid in analyzing patient pathways on multiple healthcare providers while maintaining the secrecy of patient information, thereby enabling the identification of bottlenecks and the optimization of treatment procedures [128, 109]. Supply chain management companies can benefit from

CONFINE by securely analyzing the flow of goods and information between multiple partners, thereby streamlining operations, reducing costs, and improving delivery timelines. In the manufacturing domain, CONFINE aids in examining branching workflows across independent organizations, optimizing resource allocation, and maintaining operational data secrecy. However, the acquisition of competitive advantage out of knowledge leakage must be prevented [167].

Our solution enables organizations to differentiate the sources from which the `Secure Miner` retrieves the input data. This feature allows organizations to share event logs with others while locally enriching their own dataset with additional information for their internal CONFINE protocol executions [86]. The advantage is twofold. On the one hand, this scheme lets an organization cross-link information stemming from the overarching multi-party process with internal procedure records and thus enriches the output of the `Secure Miner`. On the other hand, information that is not relevant to the external organizations is not disclosed.

A crucial aspect for facilitating the adoption of our solution in real-world settings is the availability of accessible repositories hosting and distributing variants of the `Secure Miner`, each tailored to a specific TEE technology and equipped with different process mining algorithms (as in the the work of Soriente et al. [163]). The development of trusted software for application-level TEEs requires specific Software Development Kits (SDKs), which may pose a significant integration burden for organizations lacking specialized expertise. By relying on such repositories, organizations could select the `Secure Miner` trusted application that best matches their intended process mining tasks and infrastructural constraints. At the same time, we acknowledge that this approach introduces potential risks, including malicious modifications to the `Secure Miner` source code before its deployment in a TEE. Relying on a trusted third-party hosting services for distributing verified versions of the `Secure Miner`, and counterchecking the associated measurements before startup through code-review practices can mitigate these risks.

IT engineers responsible for implementing the theoretical principles of CONFINE can instantiate the framework by following the steps outlined below. (1) They analyze the privacy requirements of the involved business entities and define their role within the framework (i.e., provisioner, miner, or both). (2) Organizations intending to participate in the CONFINE framework as miners select a TEE technology and equip themselves with a TEE-enabled processor. Cloud-based solutions may serve as an alternative for organizations lacking the necessary hardware. Different organizations may adopt distinct TEE implementations, provided that each conform to the TEE profile defined in Sect. 5.3. With such a heterogeneous setting, the specific TEE technology chosen by each organization should be known to the others to ensure

interoperability. This is especially relevant for attestations, since they require the correct handling of different attestation formats and verification services. (3) Process mining vendors select the algorithms to be integrated into their `Secure Miner`s and either obtain a verified distribution of the corresponding trusted application (as discussed above) or build their own implementation. (4) The organizations exchange the credentials that are necessary for executing the CONFINE protocol, such as the `Provisioner`s' web references, organizational credentials, and the `Secure Miner`s' measurements (which depend on the specific process mining algorithms integrated into each instance). (5) Finally, the involved organizations can start up their `Secure Miner` and `Provisioner` components to execute the CONFINE protocol.

Chapter 6

Conclusion and Future Research Directions

In this concluding chapter, we draw together the main findings of the thesis and outline future work endeavors that start therefrom.

6.1 Main findings

The thesis pursued the guiding research objective (RO) of investigating whether declarative process specifications can serve as a unifying behavioral abstraction across modeling, data analysis, and decision making in process-aware systems.

> **RO: Declarative Specifications for Process-Aware Systems**
>
> Investigate the use of LTL_f declarative specifications as an abstraction layer for representing, verifying, and exploiting behavioral knowledge in process-aware systems.

Building on this perspective, we developed contributions that span the lifecycle of declarative behavioral reasoning. We first related imperative and declarative process representations by establishing a systematic connection between them. We then moved to the data perspective, proposing a quantitative method to assess declarative specification satisfaction over executions and to interpret the resulting measurements under process drift and change. Finally, we addressed settings in which execution data is distributed across different data sources, showing how conformance checking can be performed while preserving confidentiality over the execution logs.

The remainder of this section revisits the contributions accordingly, framing them with respect to the overarching research objective (RO) and the three research questions that instantiate it.

The first research question focuses on establishing a behavioral connection between imperative and declarative process representation paradigms:

> **RQ1: Process Representation Paradigms**
>
> How can a declarative process specification be systematically derived from an imperative process model, preserving behavioral equivalence between the two representations?

In Chapter 3 we proposed the solution to RQ1, presenting a systematic approach to translate safe and sound Workflow nets into behaviorally equivalent (bisimilar) declarative specifications, bridging a long-standing gap in the literature and thus enabling comparison and interoperability between the two process representation paradigms. Technically, the encoding is obtained in a single pass and modularly over the Workflow net, preserving runs and choice points without relying on full state-space exploration and without incurring the state-space explosion typically caused by concurrency unfolding. Remarkably, the resulting specifications rely on three LTL_f formula templates from the Declare repertoire with branching: AtMostOne, End, and AlternatePrecedence("the three spells"). This yields a compact representation that remains faithful to the original behavior. As a byproduct, we show that each safe and sound Workflow net induces a star-free regular language when transitions are taken as the alphabet, strengthening the well-known regularity of languages induced by sound Workflow nets. We validated the approach through a proof-of-concept implementation, evaluating scalability on synthetic and real-world testbeds and showcasing its applicability to process diagnostics. The foundations established on behavior modeling through declarative rules pave the path for behavioral verification. Thus, once a representation is available, it becomes natural to assess how observed executions conform to it. This motivates our second research question (RQ2), stated next.

> **RQ2: Conformance Measurement**
>
> How can the satisfaction of declarative process specifications be quantitatively assessed over observed executions?

In Chapter 4, we answered RQ2, presenting an approach to quantify the satisfaction degree of rule-based LTL_f specifications on execution traces. The approach is grounded in probabilistic models with which we have derived maximum-likelihood estimators for LTL_f formulae, RCons, and process specifications. We apply our prototype to real-world data, showing its broad range of employment. We provide experimental evidence that the Confidence of a mined specification is often below the minimum levels set for the discovery of its rules. Also, we observe that a single

measure does not give a full picture of the level of interestingness of a specification for an event log. Furthermore, we describe the effect of drifts and anomalies on the specification measurements, with insights into the measures that are more suitable for the signaling of behavioral changes over time.

A key takeaway is that reasoning at the level of the overall specification is not equivalent to aggregating rule-level assessments. Indeed, rules may interact, reinforcing or constraining each other, so that decomposed analyses can miss global effects that only emerge when the specification is considered as a whole. This distinction becomes critical when execution data is distributed across multiple independent sources, so that if each party observes only a fragment of the behavior, local measurements (or their naïve combination) may not reproduce the outcome of a holistic event log level analysis. Enabling this holistic conformance assessment in decentralized settings, however, raises confidentiality concerns, since it may require access to sensitive execution data across each individual organizational boundary. These challenges motivate our third research question (RQ3), stated next.

> **RQ3: Decentralized Data Sources**
>
> How can declarative specifications be checked against execution data that is distributed across multiple independent sources?

In Chapter 5, we answered RQ3 by introducing CONFINE, a decentralized approach to conformance checking and process mining in inter-organizational settings, where execution data is distributed across independent parties. In these scenarios, organizations are often reluctant to share raw event logs beyond their boundaries due to confidentiality, security, and compliance concerns. Existing privacy-preserving solutions frequently impose information loss or heavyweight protocols that hinder scalability. CONFINE addresses these limitations by leveraging *Trusted Execution Environments* to protect data confidentiality and code integrity before, during, and after computation, thereby enabling the secure transfer and joint analysis of unaltered process data outside the perimeter of the event log providers. Concretely, CONFINE relies on an end-to-end protocol that attests the mining components first and then securely transmits and merges encrypted event data from the different parties, finally enabling running algorithms inside TEE without exposing sensitive logs in clear. We evaluated a proof-of-concept implementation on synthetic and real-world datasets, analyzing scalability and memory usage under the hardware constraints imposed by TEE-protected memory. Overall, CONFINE demonstrates that confidentiality-preserving, log-level conformance analysis is achievable even when execution data is fragmented across independent sources.

6.2 Future work

We envision several future directions based upon our presented advancements.

A natural extension of the syntheses approach presented in Chapter 3 is the inclusion of label-mappings of the Workflow net in the declarative specifications, which would turn the constraints' semi-symbolic automata into transducers that are advantageous in conformance checking contexts. To this end, investigating the handling of silent transitions and repeated labels is a path we intend to pursue. Moreover, we seek to broaden the application of our solution to detect behavioral violations, extending support to a wider range of imperative input models on one hand, and unlocking the use of the quality measures for declarative specifications defined in Chapter 4, on imperative representations [46]. Also, we aim to investigate the correspondence between specification inconsistencies and Workflow net unsafeness and unsoundness. Another promising application lies in mixed representations combining imperative and declarative paradigms [78]. In this regard, our approach could facilitate behavioral comparisons akin to [31] and enable the construction of hybrid representations tailored to diverse scenarios [61].

Looking forward to the measurement framework presented in Chapter 4, we aim to explore the enrichment of log labeling with additional contextual data (such as patient diagnoses) akin to [160] and construct estimators that take this information into account. Specifically, a possible extension would be to model the event of satisfying a formula conditional on context via logistic regression. Also, a relevant direction for practical implementations is the assessment of measures based on the most common specification and process mining tasks. To this end, the definition of desirable propositions for the interestingness measures and the properties they should guarantee is crucial, similarly to what has been done in [166] for conformance measures in process mining. Another interesting outlook is the employment of specification measures as data features for machine learning applications, e.g., trace clustering [62]. Similarly, as hinted in Sect. 4.9, our measurement framework can be used to support process mining applications, such as drift detection (with statistically significant identification of change points and pinpointing of the sub-specifications exhibiting the most erratic behavior), or post-processing filtering for declarative process discovery techniques. A stimulating research endeavor can be spurred by the application of our technique on the field, in the context of highly flexible, dynamic system and process execution scenarios such as healthcare, as in Sect. 5.4 with the checking and monitoring of rules defined in clinical pathways [172], with extensions aimed at weighting constraints according to their compulsory or best-practice nature [20].

Regarding the CONFINE approach presented in Chapter 5, first, we aim to

enhance our solution by readjusting it to the relaxation of the underlying assumptions we made, including fair conduct by data provisioners, the absence of injected or maliciously manipulated event logs, the exchange of messages through reliable communication channels where no loss or bit corruption occurs, and the existence of a universal clock for timestamps. Our future work encompasses the integration of usage control policies that specify rules on event logs' utilization, too. We plan to design enforcement and monitoring mechanisms to achieve this goal following the principles adopted in [28, 29]. We remark that a possibly severe threat to data secrecy lies in the reconstruction of the original input information from the mining output. Keeping this aspect in mind is crucial to determine the mining algorithm to be embedded in the `Secure Miner`. Studies in this regard have been conducted, among others, in [171, 87]. Integrating the proposed recommendations with CONFINE paves the path for future investigations. Finally, our approach has the potential to seamlessly cover a wide array of process mining algorithms, ranging from discovery techniques to conformance checking and performance analysis. Showing their integrability with our approach and drawing guidelines on the use of different algorithms are research directions we plan to follow.

Another promising direction to further advance the research objective of this thesis is to extend the scope of declarative specifications to *autonomous decision making*. In reinforcement learning (RL), safety constraints are often enforced by restricting the agent's action space via mechanisms such as shielding or action masking [18, 170]. While many provably safe approaches define safety in a Markovian fashion, a wide range of realistic requirements are non-Markovian and depend on the history of events and actions. As is illustrated extensively in this thesis, logic-based process specifications provide the right formalism to capture such temporal constraints. Most logic-guided methods assume that these specifications are available a priori, an assumption that limits applicability in domains where requirements are implicit, partial, or costly to determine. As ongoing work, we are investigating how declarative safety specifications can be induced from execution data, thus leveraging specification mining techniques and subsequently operationalized as guards for agents' safe exploration and execution. Concretely, mined constraints expressed in past-time temporal logic (e.g., PLTL_f [56]) can be integrated into guards so that, at each decision step, actions that would lead to a violation of any constraint are excluded. This line of research would connect specification mining with safe RL, enabling data-driven acquisition of non-Markovian safety knowledge and its use during the learning phase. Key challenges include selecting expressive template repertoires, handling noisy and heterogeneous execution traces, extending the approach to continuous action spaces, and dealing with the need for expert knowledge.

Bibliography

[1] IEEE Standard for eXtensible Event Stream (XES) for achieving interoperability in event logs and event streams (Nov 2023). doi: 10.1109/IEEESTD.2023.10267858

[2] van der Aalst, W.M.P.: Structural characterizations of sound workflow nets. Tech. Rep. 9623, Technische Universiteit Eindhoven (1996)

[3] van der Aalst, W.M.P.: Verification of workflow nets. In: ICATPN. pp. 407–426 (1997)

[4] van der Aalst, W.M.P.: The application of petri nets to workflow management. J. Circuits Syst. Comput. **8**(1), 21–66 (1998)

[5] van der Aalst, W.M.P.: Intra-and inter-organizational process mining: Discovering processes within and between organizations. In: PoEM. pp. 1–11 (2011)

[6] van der Aalst, W.M.P.: Process mining: Overview and opportunities. ACM Trans. Manag. Inf. Syst. **3**(2), 7:1–7:17 (2012)

[7] van der Aalst, W.M.P.: Conformance Checking, pp. 243–274. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). doi: 10.1007/978-3-662-49851-4_8

[8] van der Aalst, W.M.P.: Federated process mining: Exploiting event data across organizational boundaries. In: SMDS 2021. pp. 1–7 (2021)

[9] 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)

[10] van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Comput. Sci. Res. Dev. **23**(2), 99–113 (2009)

[11] van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science - R&D **23**(2), 99–113 (2009). doi: 10.1007/s00450-009-0057-9

[12] 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)

[13] van der Aalst, W.M.P., et al.: Process mining manifesto. In: BPM Workshops. pp. 169–194 (2012)

[14] Adamo, J.: Data mining for association rules and sequential patterns - sequential and parallel algorithms. Springer New York (2001). doi: 10.1007/978-1-4613-0085-4

[15] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile. pp. 487–499. Morgan Kaufmann (1994)

[16] Almagor, S., Boker, U., Kupferman, O.: Formally reasoning about quality. J. ACM **63**(3), 24:1–24:56 (2016)

[17] Alman, A., Di Ciccio, C., Maggi, F.M., Montali, M., van der Aa, H.: Rum: Declarative process mining, distilled. In: BPM. pp. 23–29 (2021)

[18] Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI). pp. 2669–2678 (2018)

[19] Altman, N., Krzywinski, M.: Simple linear regression. Nature Methods pp. 999–1000 (2015)

[20] Amantea, I.A., Robaldo, L., Sulis, E., Governatori, G., Boella, G.: Business process modelling in healthcare and compliance management: A logical framework. FLAP **9**(4), 1131–1154 (2022)

[21] Antonino, P., Woloszyn, W.A., Roscoe, A.W.: Guardian: Symbolic validation of orderliness in SGX enclaves. In: Cloud Computing Security Workshop (CCSW@CCS) 2021. pp. 111–123. ACM (2021)

[22] Augusto, A., Conforti, R., Dumas, M., et al.: Automated discovery of process models from event logs: Review and benchmark. IEEE Trans. Knowl. Data Eng. **31**(4), 686–705 (2019)

[23] Bacchus, F., Kabanza, F.: Planning for temporally extended goals. In: AAAI/IAAI, Vol. 2. pp. 1215–1222 (1996)

[24] Back, C.O., Slaats, T., Hildebrandt, T.T., Marquard, M.: DisCoveR: accurate and efficient discovery of declarative process models. Int. J. Softw. Tools Technol. Transf. **24**(4), 563–587 (2022). doi: 10.1007/s10009-021-00616-0

[25] Bagher, K., Lai, S.: Sgx-stream: A secure stream analytics framework in sgx-enabled edge cloud. Journal of Information Security and Application **72**, 103403 (2023). doi: 10.1016/J.JISA.2022.103403

[26] Barbaro, L., Varricchione, G., Montali, M., Ciccio, C.D.: From sound workflow nets to LTL_f declarative specifications by casting three spells. In: Business Process Management Forum - BPM 2025 Forum, Seville, Spain, August 31 - September 5, 2025, Proceedings. vol. 564, pp. 3–22. Springer (2025)

[27] Barrett, J.P.: The coefficient of determination—some limitations. The American Statistician pp. 19–20 (1974)

[28] Basile, D., Di Ciccio, C., Goretti, V., Kirrane, S.: Blockchain based resource governance for decentralized web environments. Frontiers in Blockchain p. 1141909 (2023)

[29] Basile, D., Di Ciccio, C., Goretti, V., Kirrane, S.: A blockchain-driven architecture for usage control in solid. In: ICDCSW. pp. 19–24 (2023)

[30] Basile, D., Goretti, V., Barbaro, L., Reijers, H.A., Di Ciccio, C.: A tee-based approach for preserving data secrecy in process mining with decentralized sources. Journal of Information Security and Applications **98**, 104381 (2026)

[31] Baumann, M.: Comparing imperative and declarative process models with flow dependencies. In: IEEESOSE 2018. pp. 63–68. IEEE Computer Society (2018)

[32] Bergmann, A., Rebmann, A., Kampik, T.: BPMN2Constraints: Breaking down BPMN diagrams into declarative process query constraints. In: BPM Demos. vol. 3469, pp. 137–141 (2023)

[33] Bickel, P., Doksum, K.: Mathematical statistics: basic ideas and selected topics, volumes I-II package. CRC Press (2015)

[34] Birkholz, H., Thaler, D., Richardson, M., Smith, N., Pan, W.: Remote attestation procedures (RATS) architecture. RFC **9334**, 1–46 (2023)

[35] Birkholz, H., Thaler, D., Richardson, M., et al.: Remote ATtestation procedureS (RATS) Architecture (2023)

[36] Buijs, J.C., van Dongen, B.F., van der Aalst, W.M.: Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. Int. J. Cooperative Inf. Syst. **23**(01), 1440001 (2014)

[37] Bulck, J.V., Minkin, M., Weisse, O., Genkin, D., Kasikci, B., et al.: Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In: USENIX 2018. USENIX Association (2018)

[38] Burattin, A., Guizzardi, G., Maggi, F.M., Montali, M.: Fifty shades of green: How informative is a compliant process trace? In: CAiSE. pp. 611–626 (2019)

[39] Busch, K., Kampik, T., Leopold, H.: xSemAD: Explainable semantic anomaly detection in event logs using sequence-to-sequence models. In: BPM. vol. 14940, pp. 309–327 (2024)

[40] Cachin, C., Guerraoui, R., Rodrigues, L.E.T.: Introduction to Reliable and Secure Distributed Programming (2. ed.). Springer (2011)

[41] Camacho, A., Triantafillou, E., Muise, C., Baier, J., McIlraith, S.: Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In: AAAI. pp. 3716–3724 (2017)

[42] Cao, Z., Tian, Y., Le, T., Lo, D.: Rule-based specification mining leveraging learning to rank. Autom. Softw. Eng. **25**(3), 501–530 (2018)

[43] Casella, G., Berger, R.L.: Statistical inference. Cengage Learning (2021)

[44] Cecconi, A., De Giacomo, G., Di Ciccio, C., Maggi, F., Mendling, J.: Measuring the interestingness of temporal logic behavioral specifications in process mining. Information Systems p. 101920 (2021)

[45] Cecconi, A., Di Ciccio, C., De Giacomo, G., Mendling, J.: Interestingness of traces in declarative process mining: The Janus LTLp_f approach. In: BPM. pp. 121–138 (2018)

[46] Cecconi, A., Barbaro, L., Ciccio, C.D., Senderovich, A.: Measuring rule-based ltlf process specifications: A probabilistic data-driven approach. Inf. Syst. **120**, 102312 (2024)

[47] Cecconi, A., Di Ciccio, C., De Giacomo, G., Mendling, J.: Interestingness of traces in declarative process mining: The Janus LTLpf approach. In: BPM. pp. 121–138 (Sep 2018)

[48] Cecconi, A., Di Ciccio, C., Senderovich, A.: Measurement of rule-based ltlf declarative process specifications. In: ICPM. pp. 96–103. IEEE (2022)

[49] Claes, J., Poels, G.: Merging event logs for process mining: A rule based merging method and rule suggestion algorithm. Expert Syst. Appl. **41**(16), 7291–7306 (2014)

[50] Corea, C., Kuhlmann, I., Thimm, M., Grant, J.: Paraconsistent reasoning for inconsistency measurement in declarative process specifications. Inf. Syst. **122**, 102347 (2024). doi: 10.1016/J.IS.2024.102347, https://doi.org/10.1016/j.is.2024.102347

[51] Cosma, V.P., Hildebrandt, T.T., Slaats, T.: Transforming dynamic condition response graphs to safe petri nets. In: PETRI NETS 2023. Lecture Notes in Computer Science, vol. 13929, pp. 417–439. Springer (2023)

[52] Costan, V., Devadas, S.: Intel SGX explained. Cryptology ePrint Archive (2016)

[53] Cramer, R., Damgård, I., Nielsen, J.B.: Secure Multiparty Computation and Secret Sharing. Cambridge University Press (2015)

[54] De Giacomo, G., Vardi, M.: Linear temporal logic and linear dynamic logic on finite traces. In: IJCAI. pp. 854–860 (2013)

[55] De Giacomo, G., De Masellis, R., Montali, M.: Reasoning on LTL on finite traces: Insensitivity to infiniteness. In: AAAI. pp. 1027–1033 (2014)

[56] De Giacomo, G., Di Stasio, A., Fuggitti, F., Rubin, S.: Pure-past linear temporal and dynamic logic on finite traces. In: IJCAI. pp. 4959–4965. ijcai.org (2020)

[57] De Giacomo, G., Maggi, F., Marrella, A., Sardiña, S.: Computing trace alignment against declarative process models through planning. In: ICAPS. pp. 367–375 (2016)

[58] De Giacomo, G., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: IJCAI. pp. 854–860 (2013)

[59] De Leoni, M., Maggi, F., van der Aalst, W.: An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data. Inf. Syst. **47**, 258–277 (2015). doi: 10.1016/j.is.2013.12.005

[60] De Leoni, M., Maggi, F.M., van der Aalst, W.M.P.: Aligning event logs and declarative process models for conformance checking. In: BPM 2012. vol. 7481, pp. 82–97 (2012)

[61] De Smedt, J., De Weerdt, J., Vanthienen, J., et al.: Mixed-paradigm process modeling with intertwined state spaces. Bus. Inf. Syst. Eng. **58**(1), 19–29 (2016)

[62] De Weerdt, J.: Trace clustering. In: Encyclopedia of Big Data Technologies (2019)

[63] De Weerdt, J., Wynn, M.T.: Foundations of process event data. In: Process Mining Handbook, pp. 193–211. Springer (2022)

[64] Desel, J., Reisig, W.: Place/transition Petri Nets, pp. 122–173 (1998)

[65] Di Ciccio, C., Mecella, M.: On the discovery of declarative control flows for artful processes. ACM Trans. Manag. Inf. Syst. **5**(4), 24:1–24:37 (2015)

[66] Di Ciccio, C., Bernardi, M.L., Cimitile, M., Maggi, F.M.: Generating event logs through the simulation of Declare models. In: EOMAS@CAiSE. pp. 20–36 (2015). doi: 10.1007/978-3-319-24626-0_2

[67] Di Ciccio, C., Maggi, F.M., Mendling, J.: Efficient discovery of Target-Branched Declare constraints. Information Systems **56**, 258–283 (Mar 2016)

[68] Di Ciccio, C., Maggi, F.M., Montali, M., et al.: Resolving inconsistencies and redundancies in declarative process models. Information Systems **64**, 425–446 (Mar 2017)

[69] Di Ciccio, C., Maggi, F.M., Montali, M., et al.: On the relevance of a business constraint to an event log. Information Systems **78**, 144–161 (Nov 2018)

[70] Di Ciccio, C., Marrella, A., Russo, A.: Knowledge-intensive Processes: Characteristics, requirements and analysis of contemporary approaches. J. Data Semantics **4**(1), 29–57 (2015). doi: 10.1007/s13740-014-0038-4

[71] Di Ciccio, C., Montali, M.: Declarative process specifications: Reasoning, discovery, monitoring. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook, pp. 108–152. Springer (2022)

[72] Di Ciccio, C., Montali, M.: Declarative Process Specifications: Reasoning, Discovery, Monitoring, pp. 108–152. Springer (Jun 2022), open access

[73] Donadello, I., Riva, F., Maggi, F.M., Shikhizada, A.: Declare4py: A python library for declarative process mining. In: Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Trackco-located with International Conference on Business Process Management (BPM) 2022. CEUR Workshop Proceedings, vol. 3216, pp. 117–121. CEUR-WS.org (2022)

[74] van Dongen, B.: BPI Challenge 2012. 4TU.ResearchData (Apr 2012)

[75] van Dongen, B.: BPI Challenge 2014. 4TU.ResearchData (Apr 2014)

[76] van Dongen, B.: BPI Challenge 2015. 4TU.ResearchData (May 2015)

[77] van Dongen, B.: Bpi challenge 2015 municipality 5 (2015)

[78] van Dongen, B., De Smedt, J., Di Ciccio, C., Mendling, J.: Conformance checking of mixed-paradigm process models. Inf. Syst. **102**, 101685 (2021)

[79] Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, Second Edition. Springer (2018)

[80] Dumas, M., Rosa, M.L., Mendling, J., et al.: Fundamentals of Business Process Management, Second Edition (2018)

[81] Dwyer, M., Avrunin, G., Corbett, J.: Patterns in property specifications for finite-state verification. In: ICSE. pp. 411–420 (1999)

[82] Elkoumy, G., Fahrenkrog-Petersen, S.A., et al.: Secure multi-party computation for inter-organizational process mining. In: BPMDS/EMMSAD. pp. 166–181 (2020)

[83] Elkoumy, G., Fahrenkrog-Petersen, S.A., et al.: Shareprom: A tool for privacy-preserving inter-organizational process mining. In: BPM (PhD/Demos). pp. 72–76 (2020)

[84] Fahland, D., Mendling, J., Reijers, H.A., et al.: Declarative versus imperative process modeling languages: The issue of maintainability. In: BPM Workshops, 2009. vol. 43, pp. 477–488 (2009)

[85] Fahrenkrog-Petersen, S.A., van der Aa, H., Weidlich, M.: PRETSA: event log sanitization for privacy-aware process discovery. In: International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24-26, 2019. pp. 1–8 (2019)

[86] Fahrenkrog-Petersen, S.A., van der Aa, H., Weidlich, M.: Optimal event log sanitization for privacy-preserving process mining. Data Knowl. Eng. **145**, 102175 (2023)

[87] Fahrenkrog-Petersen, S.A., Kabierski, M., van der Aa, H., Weidlich, M.: Semantics-aware mechanisms for control-flow anonymization in process mining. Inf. Syst. **114**, 102169 (2023)

[88] Fei, S., Yan, Z., Ding, W., Xie, H.: Security vulnerabilities of SGX and countermeasures: A survey. ACM Computing Surveys **54**(6), 126:1–126:36 (2022). doi: 10.1145/3456631

[89] Feller, W.: An introduction to probability theory and its applications. Wiley (1957)

[90] Fionda, V., Greco, G.: The complexity of LTL on finite traces: Hard and easy fragments. In: AAAI. pp. 971–977 (2016)

[91] Geng, L., Hamilton, H.: Interestingness measures for data mining: A survey. ACM Comput. Surv. **38**(3), 9 (2006)

[92] Giacomo, G.D., Felli, P., Montali, M., Perelli, G.: Hyperldlf: a logic for checking properties of finite traces process logs. In: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021. pp. 1859–1865. ijcai.org (2021)

[93] Goretti, V., Basile, D., Barbaro, L., Ciccio, C.D.: CONFINE: preserving data secrecy in decentralized process mining. In: Weerdt, J.D., Meroni, G., van der Aa, H., Winter, K. (eds.) Doctoral Consortium and Demo Track 2024 at the International Conference on Process Mining 2024 co-located with the 6th International Conference on Process Mining (ICPM 2024), Copenhagen, Denmark, October 15, 2024. CEUR Workshop Proceedings, vol. 3783. CEUR-WS.org (2024)

[94] Goretti, V., Basile, D., Barbaro, L., Ciccio, C.D.: Trusted execution environment for decentralized process mining. In: Guizzardi, G., Santoro, F.M., Mouratidis, H., Soffer, P. (eds.) Advanced Information Systems Engineering - 36th International Conference, CAiSE 2024, Limassol, Cyprus, June 3-7, 2024, Proceedings. Lecture Notes in Computer Science, vol. 14663, pp. 509–527. Springer (2024)

[95] Gorla, D.: Towards a unified approach to encodability and separation results for process calculi. Inf. Comput. **208**(9), 1031–1053 (2010)

[96] Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In: Alonso, G., Dadam, P.,

Rosemann, M. (eds.) Business Process Management. pp. 328–343. Springer Berlin Heidelberg (2007)

[97] Guzzo, A., Rullo, A., Vocaturo, E.: Process mining applications in the healthcare domain: A comprehensive review. WIREs Data Mining Knowl. Discov. **12**(2) (2022). doi: 10.1002/widm.1442

[98] Hämäläinen, W., Webb, G.: A tutorial on statistically sound pattern discovery. Data Min. Knowl. Discov. **33**(2), 325–377 (2019)

[99] Helble, S.C., Kretz, I.D., Loscocco, P.A., Ramsdell, J.D., Rowe, P.D., Alexander, P.: Flexible mechanisms for remote attestation. ACM Transactions on privacy and security **24**(4), 29:1–29:23 (2021)

[100] Hernandez-Resendiz, J.D., Tello-Leal, E., Marin-Castro, H.M., et al.: Merging event logs for inter-organizational process mining. In: New Perspectives on Enterprise Decision-Making Applying Artificial Intelligence Techniques, pp. 3–26. Springer (2021)

[101] Hildebrandt, T.T., Mukkamala, R.R.: Declarative event-based workflow as distributed Dynamic Condition Response graphs. In: PLACES 2010. vol. 69, pp. 59–73 (2010)

[102] Hodkinson, I., Reynolds, M.: Separation - past, present, and future. In: We Will Show Them! (2). pp. 117–142 (2005)

[103] Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation (3rd Edition). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2006)

[104] Huang, Z., Dong, W., Ji, L., Yin, L., Duan, H.: On local anomaly detection and analysis for clinical pathways. Artif. Intell. Medicine **65**(3), 167–177 (2015)

[105] Iacometta, C., Di Ciccio, C.: Declarative process mining with minerful, reloaded. In: CEUR Workshop Proceedings. vol. 4032, pp. 1–5. CEUR WS (2025)

[106] Ip, S., Xue, J.: A multivariate regression view of multi-label classification. Tech. rep., University College London (2015)

[107] Jans, M., Hosseinpour, M.: How active learning and process mining can act as continuous auditing catalyst. Int. J. Accounting Inf. Systems **32**, 44–58 (2019)

[108] Jauernig, P., Sadeghi, A.R., Stapf, E.: Trusted execution environments: Properties, applications, and challenges. IEEE Secur. Priv. **18**(2), 56–60 (2020)

[109] Jeon, J., Kim, J., Shin, M., Kim, M.: A blockchain-based trust model for supporting collaborative healthcare data management. Computer Systems Science and Engineering **46**(3), 3403–3421 (2023)

[110] Kamil, A., Lowe, G.: Understanding abstractions of secure channels. In: FAST. pp. 50–64 (2010)

[111] Koch, N., Kraus, A.: The expressive power of UML-based web engineering. In: IWWOST02. vol. 16, pp. 40–41 (2002)

[112] Kossiakoff, A., Sweet, W.N., Seymour, S.J., Biemer, S.M.: Systems engineering principles and practice, vol. 83. John Wiley & Sons (2011)

[113] Kunze, M., Weske, M.: Behavioural Models - From Modelling Finite Automata to Analysing Business Processes. Springer (2016)

[114] Kupferman, O., Vardi, M.: Vacuity detection in temporal model checking. Int. J. Softw. Tools Technol. Transf. **4**(2), 224–233 (2003)

[115] Lahijanian, M., Almagor, S., Fried, D., Kavraki, L., Vardi, M.: This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction. In: AAAI. pp. 3664–3671 (2015)

[116] Le, T.B., Lo, D.: Beyond support and confidence: Exploring interestingness measures for rule-based specification mining. In: SANER. pp. 331–340 (2015). doi: 10.1109/SANER.2015.7081843

[117] 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 2013. vol. 171, pp. 66–78 (2013)

[118] Legay, A., Lukina, A., Traonouez, L., Yang, J., Smolka, S., Grosu, R.: Statistical model checking. In: Computing and Software Science - State of the Art and Perspectives, pp. 478–504 (2019)

[119] Lemieux, C., Park, D., Beschastnikh, I.: General LTL specification mining (T). In: ASE. pp. 81–92 (2015)

[120] Lemieux, C., Park, D., Beschastnikh, I.: General LTL specification mining (T). In: ASE. pp. 81–92 (2015)

[121] Lenca, P., Meyer, P., Vaillant, B., Lallich, S.: On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid. Eur. J. Oper. Res. **184**(2), 610–626 (2008)

[122] Leo, S., Foglie, A.D., Barbaro, L., Marangone, E., Panetta, I.C., Ciccio, C.D.: Transforming credit guarantee schemes with distributed ledger technology. CoRR **abs/2404.19555** (2024)

[123] Leo, S., Foglie, A.D., Barbaro, L., Marangone, E., Panetta, I.C., Ciccio, C.D.: Blink: Blockchain-linked network for credit guarantee institutions (Feb 2026), to Appear in Financial Innovation Journal

[124] de Leoni, M., Mannhardt, F.: Road traffic fine management process (Feb 2015)

[125] Li, J., Zhang, L., Pu, G., Vardi, M.Y., He, J.: Ltlf satisfiability checking. In: ECAI. Frontiers in Artificial Intelligence and Applications, vol. 263, pp. 513–518. IOS Press (2014). doi: 10.3233/978-1-61499-419-0-513

[126] Lichtenstein, O., Pnueli, A., Zuck, L.D.: The glory of the past. In: Logics of Programs. pp. 196–218 (1985)

[127] Liu, C., Li, Q., Zhao, X.: Challenges and opportunities in collaborative business process management: Overview of recent advances and introduction to the special issue. Inf. Syst. Front. **11**, 201–209 (2009)

[128] Liu, C., Li, H., Zhang, S., Cheng, L., Zeng, Q.: Cross-department collaborative healthcare process model discovery from event logs. IEEE Transactions on Automation Science and Engineering **20**(3), 2115–2125 (2023)

[129] Maaradji, A., Dumas, M., Rosa, M.L., Ostovar, A.: Detecting sudden and gradual drifts in business processes from execution traces. IEEE Trans. Knowl. Data Eng. **29**(10), 2140–2154 (2017)

[130] Maggi, F.M., Mooij, A.J., van der Aalst, W.M.P.: User-guided discovery of declarative process models. In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11-15, 2011, Paris, France. pp. 192–199. IEEE (2011). doi: 10.1109/CIDM.2011.5949297

[131] Maggi, F.M., Mooij, A.J., van der Aalst, W.M.P.: User-guided discovery of declarative process models. In: CIDM. pp. 192–199 (2011). doi: 10.1109/CIDM.2011.5949297

[132] Maggi, F., Montali, M., Peñaloza, R.: Temporal logics over finite traces with uncertainty. In: AAAI. pp. 10218–10225 (2020)

[133] Mannhardt, F.: Sepsis cases - event log (Dec 2016)

[134] Mannhardt, F.: Sepsis cases - event log (2016)

[135] Mei, B., Xia, S., Wang, W., Lin, D.: Cabin: Confining untrusted programs within confidential vms. In: Information and Communications Security (ICICS) 2024. Lecture Notes in Computer Science, vol. 15056, pp. 165–184. Springer (2024)

[136] Ménétrey, J., Göttel, C., Khurshid, A., Pasin, M., Felber, P., Schiavoni, V., Raza, S.: Attestation mechanisms for trusted execution environments demystified. In: Distributed Applications and Interoperable Systems (DAIS) 2022. vol. 13272, pp. 95–113 (2022)

[137] Morin, E.: La méthode: la nature de la nature. Média Diffusion (2013)

[138] Müller, M., Ostern, N., Koljada, et al.: Trust mining: analyzing trust in collaborative business processes. IEEE Access pp. 65044–65065 (2021)

[139] Müller, M., Simonet-Boulogne, A., Sengupta, S., Beige, O.: Process mining in trusted execution environments: Towards hardware guarantees for trust-aware inter-organizational process analysis. In: ICPM. pp. 369–381 (2021)

[140] Munoz-Gama, J., Martin, N., Fernández-Llatas, C., et al.: Process mining for healthcare: Characteristics and challenges. J. Biomed. Informatics **127**, 103994 (2022)

[141] Murdock, K., Oswald, D.F., Garcia, F.D., et al.: Plundervolt: Software-based fault injection attacks against intel SGX. In: IEEE Symposium on Security and Privacy (SP) 2020. pp. 1466–1482. IEEE (2020)

[142] Ouyang, C., Dumas, M., van der Aalst, W.M.P., et al.: From business process models to process-oriented software systems. ACM Trans. Softw. Eng. Methodol. **19**(1), 2:1–2:37 (2009)

[143] Pentland, B.T., Recker, J., Kim, I.: Capturing reality in flight? empirical tools for strong process theory. In: Kim, Y.J., Agarwal, R., Lee, J.K. (eds.) Proceedings of the International Conference on Information Systems - Transforming Society with Digital Innovation, ICIS 2017, Seoul, South Korea, December 10-13, 2017. Association for Information Systems (2017)

[144] Pesic, M., Bosnacki, D., van der Aalst, W.: Enacting declarative languages using LTL: avoiding errors and improving performance. In: SPIN. pp. 146–161 (2010)

[145] Pesic, M.: Constraint-based Workflow Management Systems: Shifting Control to Users. Ph.D. thesis, TU Eindhoven (10 2008)

[146] Petri, C.A.: Kommunikation mit automaten (1962), https://api.semanticscholar.org/CorpusID:117254333

[147] Pichler, P., Weber, B., Zugal, S., et al.: Imperative versus declarative process modeling languages: An empirical investigation. In: BPM Workshops 2011. vol. 99, pp. 383–394 (2011)

[148] Piribauer, J., Baier, C., Bertrand, N., Sankur, O.: Quantified linear temporal logic over probabilistic systems with an application to vacuity checking. In: CONCUR. pp. 7:1–7:18 (2021)

[149] Pnueli, A.: The temporal logic of programs. In: FOCS. pp. 46–57 (1977)

[150] Polato, M.: Dataset belonging to the help desk log of an italian company (Jul 2017)

[151] Polyvyanyy, A., Solti, A., Weidlich, M., Di Ciccio, C., Mendling, J.: Monotone precision and recall measures for comparing executions and specifications of dynamic systems. ACM Trans. Softw. Eng. Methodol. **29**(3), 17:1–17:41 (2020). doi: 10.1145/3387909

[152] Prescher, J., Di Ciccio, C., Mendling, J.: From declarative processes to imperative models. In: SIMPDA 2014. vol. 1293, pp. 162–173 (2014)

[153] Rebmann, A., Kampik, T., Corea, C., van der Aa, H.: Mining constraints from reference process models for detecting best-practice violations in event log. CoRR **abs/2407.02336** (2024)

[154] Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems (reprint). Commun. ACM **26**(1), 96–99 (1983)

[155] Rocha, E.G., van Zelst, S.J., van der Aalst, W.M.P.: Mining behavioral patterns for conformance diagnostics. In: BPM 2024. vol. 14940, pp. 291–308 (2024)

[156] Rovani, M., Maggi, F.M., de Leoni, M., van der Aalst, W.M.P.: Declarative process mining in healthcare. Expert Syst. Appl. **42**(23), 9236–9251 (2015). doi: 10.1016/j.eswa.2015.07.040

[157] Rozinat, A., van der Aalst, W.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**(1), 64–95 (2008). doi: https://doi.org/10.1016/j.is.2007.07.001

[158] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach (4th Edition). Pearson (2020)

[159] Sabt, M., Achemlal, M., Bouabdallah, A.: Trusted execution environment: What it is, and what it is not. In: 2015 IEEE TrustCom/BigDataSE/ISPA. pp. 57–64 (2015)

[160] Schönig, S., Di Ciccio, C., Maggi, F.M., Mendling, J.: Discovery of multi-perspective declarative process models. In: ICSOC. pp. 87–103 (2016). doi: 10.1007/978-3-319-46295-0_6

[161] Schützenmeier, N., Corea, C., Delfmann, P., Jablonski, S.: Efficient computation of behavioral changes in declarative process models. In: BPMDS/EMMSAD@CAiSE. Lecture Notes in Business Information Processing, vol. 479, pp. 136–151. Springer (2023)

[162] Sinik, T., Beerepoot, I., Reijers, H.A.: A peek into the working day: Comparing techniques for recording employee behaviour. In: RCIS. vol. 476, pp. 343–359 (2023)

[163] Soriente, C., Karame, G., Li, W., Fedorov, S.: Replicatee: Enabling seamless replication of SGX enclaves in the cloud. In: IEEE European Symposium on Security and Privacy (EuroS&P) 2019. pp. 158–171. IEEE (2019)

[164] Steeman, W.: BPI challenge 2013, incidents (2013)

[165] Steeman, W.: BPI Challenge 2013 (Apr 2014)

[166] 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)

[167] Tan, K.H., Wong, W.P., Chung, L.: Information and knowledge leakage in supply chain. Inf. Syst. Frontiers **18**(3), 621–638 (2016)

[168] Tan, P.N., Kumar, V., Srivastava, J.: Selecting the right objective measure for association analysis. Information Systems **29**(4), 293 – 313 (2004), knowledge Discovery and Data Mining (KDD 2002)

[169] Thomas, S.A.: SSL and TLS Essentials: Securing the Web. Wiley (2000)

[170] Varricchione, G., Alechina, N., Dastani, M., Giacomo, G.D., Logan, B., Perelli, G.: Pure-past action masking. In: Wooldridge, M.J., Dy, J.G., Natarajan, S. (eds.) Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada. pp. 21646–21655. AAAI Press (2024)

[171] von Voigt, S.N., Fahrenkrog-Petersen, S.A., et al.: Quantifying the re-identification risk of event logs for process mining - Empiricial evaluation paper. In: CAiSE. pp. 252–267 (2020)

[172] Weerdt, J.D., Caron, F., Vanthienen, J., Baesens, B.: Getting a grasp on clinical pathway data: An approach based on process mining. In: Emerging Trends in Knowledge Discovery and Data Mining - PAKDD 2012 International Workshops: DMHM, GeoDoc, 3Clust, and DSDM, Kuala Lumpur, Malaysia, May 29 - June 1, 2012, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7769, pp. 22–35. Springer (2012)

[173] Weijters, A.J.M.M., van der Aalst, W.M.P., Alves De Medeiros, A.K.: Process mining with the HeuristicsMiner algorithm (2006)

[174] Westergaard, M., Slaats, T.: Mixing paradigms for more comprehensible models. In: BPM 2013. vol. 8094, pp. 283–290 (2013)

[175] Yang, J., Evans, D., Bhardwaj, D., Bhat, T., Das, M.: Perracotta: mining temporal API rules from imperfect traces. In: ICSE. pp. 282–291 (2006)

[176] Yasmin, F.A., Bukhsh, F.A., de Alencar Silva, P.: Process enhancement in process mining: A literature review. In: Proceedings of the 8th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2018). CEUR Workshop Proceedings, vol. 2270, pp. 65–72. CEUR-WS.org (2018)

[177] Yeshchenko, A., Di Ciccio, C., Mendling, J., Polyvyanyy, A.: Comprehensive process drift detection with visual analytics. In: ER. pp. 119–135 (2019)

[178] Yeshchenko, A., Di Ciccio, C., Mendling, J., Polyvyanyy, A.: Visual drift detection for event sequence data of business processes. IEEE Trans. Vis. Comput. Graph. **28**(8), 3050–3068 (2022)

[179] Yeshchenko, A., Mendling, J., Di Ciccio, C., Polyvyanyy, A.: VDD: A visual drift detection system for process mining. In: ICPM Doctoral Consortium / Tools. pp. 31–34 (2020)

[180] Zhao, C., Zhao, S., Zhao, M., et al.: Secure multi-party computation: Theory, practice and applications. Inf. Sci. **476**, 357–372 (2019)