Interactive process mining

Interactive Process Mining

P.M. (Alok) Dixit

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Dixit, P.M. (Alok)

Interactive Process Mining by P.M. (Alok) Dixit.
Eindhoven: Technische Universiteit Eindhoven, 2019. Proefschrift.

Cover design by Remco Wetzels

A catalogue record is available from the Eindhoven University of Technology Library

ISBN 978-90-386-4766-1.

Keywords: interactive process mining, process discovery, human-in-the-loop, business process management, process data quality, event log repair, process model repair, fast conformance analysis

The work in this thesis has been sponsored by the Data Science flagship collaboration between TU/e and Philips.

Printed by IPSKAMP printing, Enschede

Interactive Process Mining

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de rector magnificus prof.dr.ir. F.P.T. Baaijens, voor een commissie aangewezen door het College voor Promoties, in het openbaar te verdedigen op
dinsdag 19 juni 2019 om 13:30 uur

door

Prabhakar (Alok) Madhukar Dixit

geboren te Ponda, Goa, India

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof.dr.ir. B. Koren
1e promotor:	prof.dr.ir. W.M.P. van der Aalst
2e promotor:	prof.dr.ir. J.J. van Wijk
copromotor:	dr.ir. H.M.W. Verbeek
leden:	prof.dr. J. Esparza (Technische Universität München)
	prof.dr. M. Weidlich (Humboldt-Universität zu Berlin)
	dr.ir. M.C. Willemsen
overige leden:	dr. J. Korst (Philips research)

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

To my parents. . .

Abstract

A process is a series of actions performed in order to achieve a particular task or goal. Processes are omnipresent, for example, a manufacturing process which deals with the production of some goods, a loan application process in a bank, a hospital visit of a patient. The human understanding of these processes is usually facilitated by representing them as intuitive graphical models. Typically a domain expert, who has a high level overview of the end-to-end execution of a process, constructs the expected process model. Since these modeling notations offer intuitive visualization of processes, the resulting models have proved to be valuable artifacts to enable communication of complex knowledge in a comprehensible way across people from dissimilar backgrounds. The process model as described by a domain expert is the best guess, of how a process acts, or should act. However, reality may not always conform to this expected behavior of the process.

The execution histories of processes, called *event logs*, can be easily extracted from the corresponding information systems. For example, the event logs of an administrative process can be extracted from an Enterprise Resource Planning (ERP) system, or the event logs of a careflow process in a hospital can be extracted from the hospitals Electronic Health Record (EHR) system. These event logs provide a view on the execution of processes in reality and can thus be used to discover process models when the process model is unknown, or to analyze the behavior of the process in reality. This branch of science which uses event logs to analyze the process behavior is known as *process mining*.

Process modeling and process mining are typically on the extreme ends of a spectrum. On one end, we have the process mining techniques, which analyze event logs in order to *automatically* deduce insights, for example discovering process models. The user has very little influence when using such techniques, and usually, it is not possible to incorporate the domain knowledge during process mining. On the other end, we have the traditional process modeling tools which are completely user driven and use no historical evidence from the event logs during process modeling. In the manual process modeling scenario, the user may miss out on important real-life execution information that may be present in the event logs. Whereas the automated process mining techniques may produce results which may be inaccurate and incomprehensible due to noisy and/or incomplete event logs.

In this thesis, we address this gap between the traditional *human-driven* process modeling world, with the *data-driven* process mining world, by developing techni-

ques that enable interactive process mining. We identify and primarily address four research goals in this thesis:

- *Interactive event log repair*: Addresses the issue of repairing the source of process mining analysis, i.e., the event log, with the domain knowledge of an informed user.
- *Interactive process model construction*: Addresses the issue of modeling/discovering process models interactively by using both information derived from the event log and the expertise of a user. Techniques are developed to assist the user in decision-making by
  - providing fast conformance analysis between the event log and the process being modeled.
  - providing recommendations to enable semi-automated process discovery by combining automated and interactive approaches for process modeling.

  Furthermore, we aim to provide formal guarantees for the resulting process model. To ensure the soundness of the constructed process model, we explore the use of liveness and boundedness preserving synthesis rules for free-choice Petri nets.
- *Interactive process model repair*: Addresses the issue of interactively repairing a pre-existing process model by using the domain knowledge and information derived from the event log.
- *Interactive process analytics*: Addresses the issue of interactively exploring real-life execution of a process based on the process model and the event log, in order to evaluate the possible compliance and performance oriented issues.

The proposed techniques have all been implemented and evaluated using several real-life scenarios involving real event data and/or process experts.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, process mining has emerged as an important discipline that has its roots in process modeling and data mining. Typically process mining approaches are automated, with little or no influence from the user. In this thesis, we focus on addressing this gap and providing approaches that enable interactive process mining. In Section 1.1, we first discuss the relevance of process science and data science in a broader sense, and the omnipresence of processes and data in everyday scenarios. In Section 1.2, we discuss the traditional ways of process modeling, which constitute the core of process science. In Section 1.3, we discuss how process mining approaches process science-based questions in a data-driven way. In Section 1.4, we discuss the missing link between process modeling based approaches and process mining based approaches, i.e., the need for interactive capabilities in a process mining setting. We identify four areas within process mining which serve as the four main goals of the thesis. In Section 1.5, we discuss the outline of the thesis, and briefly discuss the contents of the chapters that constitute to the goals discussed in Section 1.4.

1.1 Process Orientation in a Data-driven World

We live in a world where we are constantly surrounded by processes, in every aspect of work or leisure. A process is defined as a series of steps taken, or actions performed, in order to achieve a particular goal. We may not realize it, but we are constantly involved in multiple processes, either actively or passively.

Consider the case of a patient visiting a hospital. Clearly, the hospital would have some protocols, based on which, the "steps" a patient should follow in the hospital would be known to the hospital staff. Let us consider Figure 1.1, which shows the care-pathway of a patient in the hospital. When the patient named Pete walks into the hospital, he reports at the reception desk, and is admitted in the hospital by a receptionist Romy, as designated by the *1) Admission* step. Next, the receptionist may advise the patient to consult an in-house specialist in the hospital, as shown by the *2) Consult Specialist* step. The specialist Simon may request Pete to perform a couple of

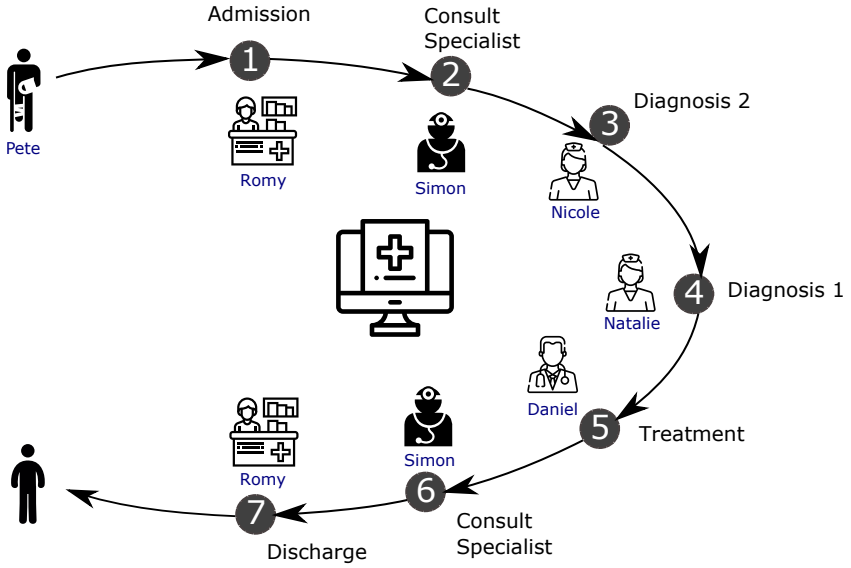


Figure 1.1: Care pathway of a patient in a hospital.

diagnostic tests, based on Pete's situation. Thus, Pete undergoes *3) Diagnosis 2* and *4) Diagnosis 1* steps, which are performed by the nurses named Nicole and Natalie from the hospital. Pete then visits a doctor for generic *5) Treatment*, and is then re-directed to the specialist for final checks. Simon performs final checks, denoted by the step *6) Consult Specialist*, and Pete is then ready for discharge. Romy at the reception prepares the necessary medication and processes the billing, after which Pete is discharged from the hospital, as denoted by *7) Discharge*.

Figure 1.1 shows the different steps that a patient followed in the hospital, which all form a part of the care-pathway process of the hospital. There is a staff member responsible for performing a particular step or task of the process. It should be noted that not all the staff members of the hospital involved in the care-pathway process, would be aware of all the steps in the care-pathway process, for example the receptionist Romy may not be aware of what should happen after *4) Diagnosis 1* is performed. However, typically, the resources involved in the process are aware of the *next* step that should be performed, immediately following the step that they were involved in. Furthermore, it is quite obvious that not all the patients would follow the exact same steps in the process, as shown in Figure 1.1. These steps would be dependent on many factors, such as the symptoms of the patient, the specific conditions that a patient may be suffering from, work load of the doctor, and so on. Therefore, the actual underlying care-pathway process would not be linear, as some patients may have additional steps taken, some steps skipped, some steps repeated etc.

The advent of digitization has lead to a transition from paper-based to information-systems-based recording of data. This has lead to huge amounts of data being recorded, leading to the phenomenon of "big data". In essence, data is collected *about*

anything, *at any time* and *at any place* [163]. In essence, this phenomena leads to the so-called Internet of Events (IoE), where events may be "life events", "machine events" or "organization events".

Clearly, the steps followed by a patient in the care-pathway process of the hospital also constitute events. Each event corresponds to the actual "task" that was performed, the "resource" that was responsible for performing the said task, the exact "time" when the task was performed, the "patient" for whom the task was performed, and so on. Such event data are typically recorded in the Hospital Information System (HIS) of the hospital, which is the central information system across all the steps of the care-pathway as shown in Figure 1.1.

The events from the HIS, could then be traced back to the actual care-pathway process, which created them in the first place. Tracing such events back to the processes could be useful in multiple ways, and facilitate better understanding of the processes that created these events. Moreover, such event data could be analyzed to improve these processes, for example, understanding what is going on in reality, changing the flow of patient, to improve the overall efficiency of the care-pathway process, and so on.

Hence, it is imperative to utilize the event data in order to understand, analyze and improve processes. There are two classical approaches for achieving this. (i) The *process-modeling approach*: which is a human-driven top-down approach. Typically, the starting point in such an approach is a hand-made process model, which is then used in combination with the event data to analyze, repair and improve processes. (ii) The *process mining approach*: which is a data-driven bottom-up approach. These approaches typically rely almost entirely on the event data in order to discover, analyze and enhance processes. In the next sections, we discuss these two approaches in detail.

1.2 Process Modeling

Traditional process management starts from hand-made process models. These process models could either be formal or informal. On one hand, the formal process models carry semantics, and can be specified in terms of executable code in a workflow management system. On the other hand, the informal process models do not have any enactable semantics, i.e., such process models are high-level process models that cannot be executed without adding additional information. The informal process models are typically vague and ambiguous. Hence, it is desirable to only focus on the executable class of process models, i.e., formal process models.

As discussed in the context of the care-pathway process, the hospital staff involved in the process, may only be aware of the steps immediately following the steps that they are involved in. Hence, typically in order to come up with an end-to-end process model describing the care-pathway of the hospital for all (or the majority) of the patients, a process analyst or a designer is appointed. Such a process analyst/designer typically interviews different resources involved in different stages of the process, and combines the extracted domain knowledge in order to construct the process model in

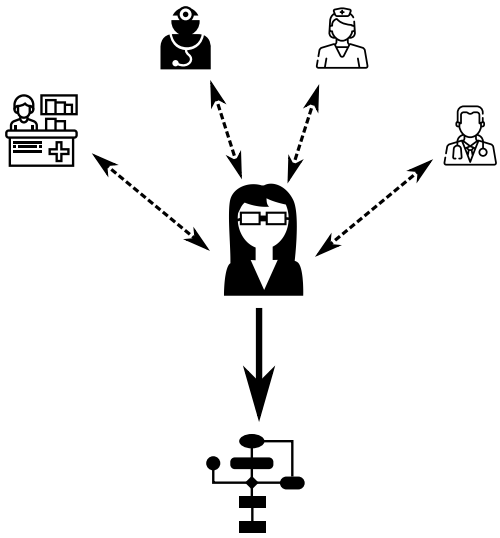


Figure 1.2: The conventional process modeling approach involves a process analyst that interviews different actors involved in the process, and consolidates all the information in order to obtain a process model.

the most suitable way, as shown in Figure 1.2.

A prominent way of modeling formal process models is using the Petri nets notation [130]. Figure 1.3 shows a simplistic example of a care-pathway process in a hospital. The process model shows the steps followed by a patient who visits the hospital, and is navigated from left-to-right. The first step is *Admission* which is represented by a *transition*, i.e., a rectangle. A *transition* is enabled, i.e., the corresponding activity (step) can occur, if each incoming place (i.e., circle) contain a so-called token. After an activity completes, the transition is *fired*, resulting in the removal of

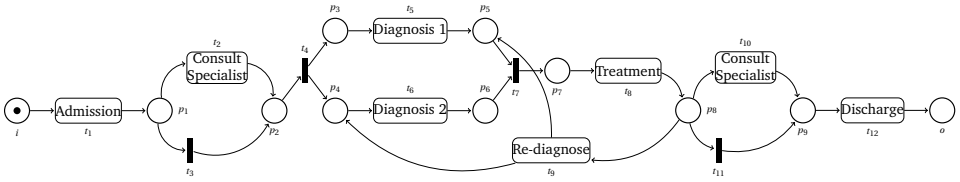


Figure 1.3: An example process model represented by a Petri net showing a simplistic care-pathway in a hospital.

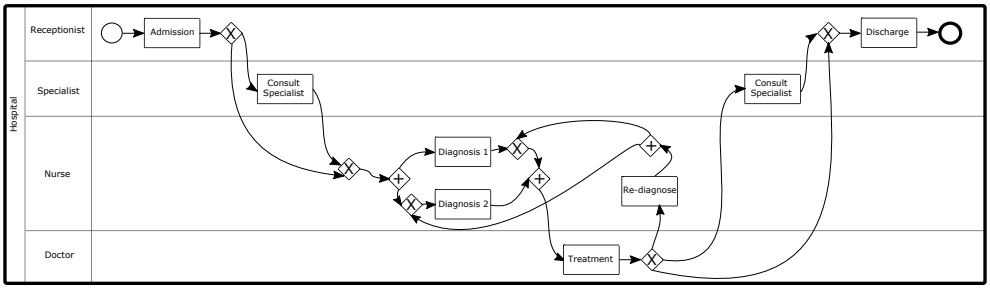


Figure 1.4: BPMN equivalent of the process model from Figure 1.3.

one token from each of the incoming places and an addition of one token to each of the outgoing places of the transition. In the Petri net from Figure 1.3, activity Admission is enabled, as the place i contains a token. Once the patient is admitted in the hospital, the next step that can be performed is *Consult Specialist* (t_2) or it can be skipped via a *silent transition* (t_3), i.e., the transition colored black representing no-behavior. The next silent transition (t_4) fires and results in a token each in places p_3 and p_4, thereby enabling both *Diagnosis 1* (t_5) and *Diagnosis 2* (t_6). Thereby, both the transitions labeled *Diagnosis 1* and *Diagnosis 2* are enabled, indicating concurrency in the process. Once both the activities are executed, i.e., transitions t_5 and t_6 fired, the following silent transition (t_7) synchronizes the process and results in a token in the place p_7. This enables *Treatment* activity (t_8). After *Treatment*, there is a token in place p_8, and hence, the process has three choices. First, the activity *Re-diagnose* (t_9) can be performed, resulting in a token in the places p_4 and p_5 and hence, eventual repetition of the activities *Diagnosis 2* and *Treatment*. The second and third options are to continue with the process, instead of looping back. In the second option, the activity *Consult Specialist* (t_{10}) is executed, and in the third option the activity *Consult Specialist* is skipped by firing the silent transition (t_{11}). This results in a token in place p_9, which would finally enable activity *Discharge* (t_{12}), executing which, the process would be terminated.

Next to Petri nets, there are other modeling notations such as BPMN, Process Trees, UML, Activity diagrams and Statecharts, EPC etc. which can be used to represent process models. For example, Figure 1.4 shows the care-pathway process from Figure 1.3 represented as a BPMN diagram with swim-lanes indicating the resources responsible for the different steps in the process.

Process models can be used in various scenarios. First and foremost, such process models are used in order to represent the *control flow*, i.e., ordering sequence of activities in a process, in a graphical manner. This is particularly useful in order to get an overview of the complete process, and facilitate discussions among stakeholders. Such graphical notations are easy for interpretation, and the complexity and the ease of understanding of the overall process model can be managed by the process designer, while constructing the process model by hand.

Formal process models can be used for the verification of processes, i.e., to ana-

lyze the possible errors in systems or process executions, for example, deadlocks in processes. Furthermore, formal process models can be used to analyze the animations of processes to play out different scenarios and thus provide feedback to the analyst. Finally such process models are also useful to document the processes. Hence, it can be argued that the process models play a key role in large organizations. However, very often, the hand-made process models are disconnected from reality. That is, there may be gaps between the expected process behavior, as depicted by a process model, and what actually happens in reality. Therefore, even though a formally verifiable process model offers rigorous analysis techniques, it may still be unreliable when compared with the actual execution of the process, and hence may not be completely useful.

Fortunately, the real executions of processes are often recorded in the information systems and can be extracted using the event data. Therefore, we can make use of such event data in order to analyze process behavior in reality. Process mining aims to do exactly this, and is discussed in detail in the next section.

1.3 Process Mining

Applications of data-science-based techniques to the field of Business Process Management resulted in the origin of process mining. Compared to the top-down approach of traditional process modeling, process mining techniques work in a bottom-up way. That is, analysis are performed starting with the data recorded during the execution of processes in reality. Figure 1.5 shows a high-level overview of various types of process mining techniques. We now discuss the components from Figure 1.5 in detail.

Event logs are the starting point for performing any process mining analysis. Event logs contain process oriented event data extracted from various information sources, such as Enterprise Resource Planning (ERP) systems, Electronic Medical Records (EMR), Hospital Information Systems (HIS) and so on. An event log contains information about events, related to a process. Table 1.1 shows an example event log. Each event minimally indicates occurrence of an activity in the process, and is related to a particular case (i.e., a process instance). Furthermore, the events belonging to a single case are ordered, typically based on the timestamps of the events. Thus, a case describes a single "run" of the process. Case 2 from the example event log of Table 1.1 indeed corresponds to the patient journey of Pete as shown in Figure 1.1. Event logs may additionally contain information about events, such as which resource was responsible for performing a particular activity, corresponding to the event. Furthermore, an event log may also contain additional case-level attributes, such as the gender of the patient etc. Using such an event log, we can then perform process mining analysis.

The first type of process mining is called *process discovery*. As the name suggests, the goal of process discovery techniques is to discover a process model solely based on the information from the event log. The α-miner was among the first process discovery techniques which takes as input an event log and produces a process model, represented by a Petri net, as the output. Most of the traditional process discovery

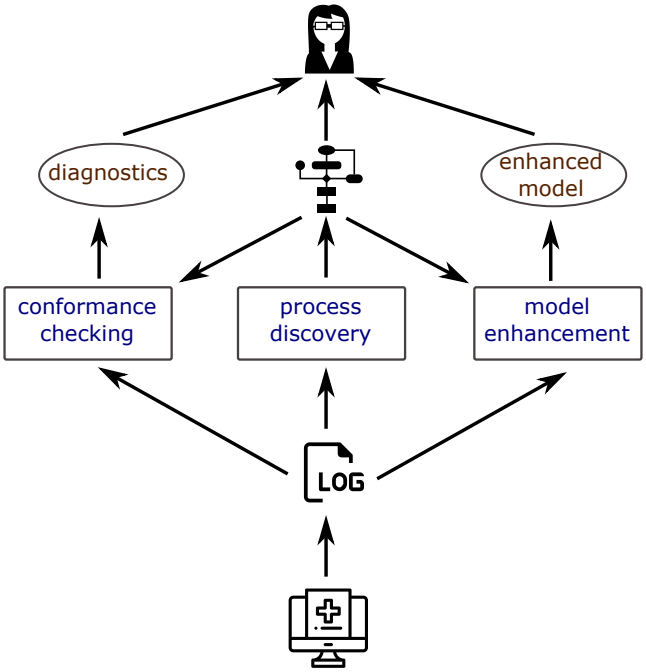


Figure 1.5: Process mining overview [161].

techniques are automated in nature, and allow for limited configuration using certain parameters. The parameters configured are either used in order to overcome shortcomings in the event log, or to specify the types of constructs and the complexity of the process model being discovered. However, once the parameters are set, the task of process discovery is completely automated, and the user has no influence over the actual discovery.

The second type of process mining is *conformance checking*. In conformance checking, a pre-existing process model is compared to the event log of the same process. Conformance checking is used to check if the reality, as represented by the event log, conforms to the model, as depicted by the process model, and vice versa. The example event log from Table 1.1 shows two cases, i.e., case 1 and 2 that conform to the process model shown in Figure 1.3. However, case 3 from the event log of Table 1.1 does not conform with the process model of Figure 1.3, as it has an occurrence of activity *Consult Specialist* before the *Treatment* activity which is not supported according to the process model of Figure 1.3. Conformance analysis is useful to identify possible compliance issues in the process model. Moreover, conformance analysis could also be useful to detect, locate and explain deviations. Furthermore, some process discovery techniques, such as genetic algorithms (for example ETM algorithm [24]) use conformance analysis for guiding the process discovery.

The third type of process mining is *enhancement*. In case of enhancement, both

Table 1.1: An example event log of a care pathway process in a hospital.

Case ID	Activity	TimeStamp	Resource	⋯
1	Admission	1-1-2004 12:34	Romy	⋯
1	Consult Specialist	1-1-2004 12:38	Simon	⋯
1	Diagnosis 1	1-1-2004 12:45	Nicole	⋯
1	Diagnosis 2	1-1-2004 12:56	Natalie	⋯
1	Treatment	1-1-2004 13:00	Daniel	⋯
1	Re-diagnose	1-1-2004 13:34	Natasha	⋯
1	Diagnosis 2	1-1-2004 13:50	Natalie	⋯
1	Treatment	1-1-2004 14:00	Daniel	⋯
1	Discharge	1-1-2004 14:40	Romy	⋯
2	Admission	1-1-2004 12:41	Romy	⋯
2	Consult Specialist	1-1-2004 12:48	Simon	⋯
2	Diagnosis 2	1-1-2004 13:48	Nicole	⋯
2	Diagnosis 1	1-1-2004 14:19	Natalie	⋯
2	Treatment	1-1-2004 14:30	Daniel	⋯
2	Consult Specialist	1-1-2004 14:49	Simon	⋯
2	Discharge	1-1-2004 15:23	Romy	⋯
3	Admission	1-1-2004 13:11	Romy	⋯
3	Consult Specialist	1-1-2004 13:51	Simon	⋯
3	Diagnosis 2	1-1-2004 14:18	Nicole	⋯
3	Diagnosis 1	1-1-2004 15:01	Natalie	⋯
3	Consult Specialist	1-1-2004 15:40	Simon	⋯
3	Treatment	1-1-2004 15:53	Daniel	⋯
3	Discharge	1-1-2004 16:02	Romy	⋯
⋮	⋮	⋮	⋮	⋱

the process model and event log are present, and the idea is to extend or improve a pre-existing process model with the information from the event log. One of the types of enhancement is *process model repair*, which aims to repair a pre-existing process model based on the data from the event log. In essence, information from conformance analysis could also be used in order to identify the possible issues and perform repair actions. Another type of enhancement is *extension*, where the idea is to add a new data-driven perspective to the process model. An example of this could be to project process performance oriented information onto a process model.

As shown, process mining techniques aim to exploit the event data in order to analyze different aspects of processes. However, in such settings, user activity is limited to analyzing the results that are already mined/discovered by process mining algorithms. That is, a user's role is limited to configuring parameters of the automated

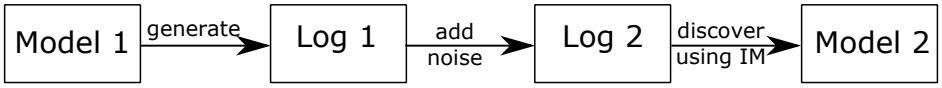


Figure 1.6: Steps followed to demonstrate the issues with automated process discovery techniques. Model 1 (Figure 1.3) is used to simulate an event log, i.e., Log 1. Random noise is added to this event log to obtain Log 2. Log 2 is used to automatically discover a process model using the inductive miner infrequent (IM) algorithm [110].

process mining techniques. Hence, it is not possible to allow the user to actively and interactively influence process mining outcomes. In the next section, we discuss in detail, the motivations for interactive process mining, and the research goals that shape this thesis.

1.4 Incorporating User Knowledge in Process Mining

It is quite evident that both the process modeling and process mining approaches have their pros and cons. The process modeling approaches typically *miss-out* on the reality, as the process that is modeled is based entirely on the knowledge of the process analyst. That is, the actors in the process may *think* they are doing something, and hence that is how the process would be modeled. However, it is not uncommon that in reality something completely different happens. Process mining covers the other angle pretty well. However, in traditional process mining scenarios, there is very limited support for a user to incorporate domain knowledge to influence process mining outcomes. In many real-life scenarios, the extracted event log may contain incomplete and/or noisy information. Since an event log is the primary source of information for the process mining scenario, the end result may be highly inaccurate.

We demonstrate one of the limitations of process mining, process discovery in particular, with the help of an example. Consider an event log generated from the process model of Figure 1.3. In order to replicate reality, we introduced random noise to such an event log. Usage of one of the prominent state-of-the-art discovery techniques (inductive miner infrequent [110]) on the noisy event log results in the process model shown in Figure 1.7. Figure 1.6 shows the set-up for obtaining the process model from Figure 1.7.

Automated discovery techniques try to extract useful information from noisy event data in order to discover a process model. However, discovery techniques still have limitations. For instance, in the discovered process model of Figure 1.7, it is possible to perform *Diagnosis* and *Consult Specialist* without performing the *Admission* activity. However, this does not sound realistic. The human created process model from Figure 1.3 performs better (in terms of quality metrics such as *fitness* and *precision*) than the automatically discovered process model from Figure 1.7 on the noisy event log (Log 2 of Figure 1.6). Therefore, a human created process model may still outperform a discovered model when compared with the event log using metrics like fitness, pre-

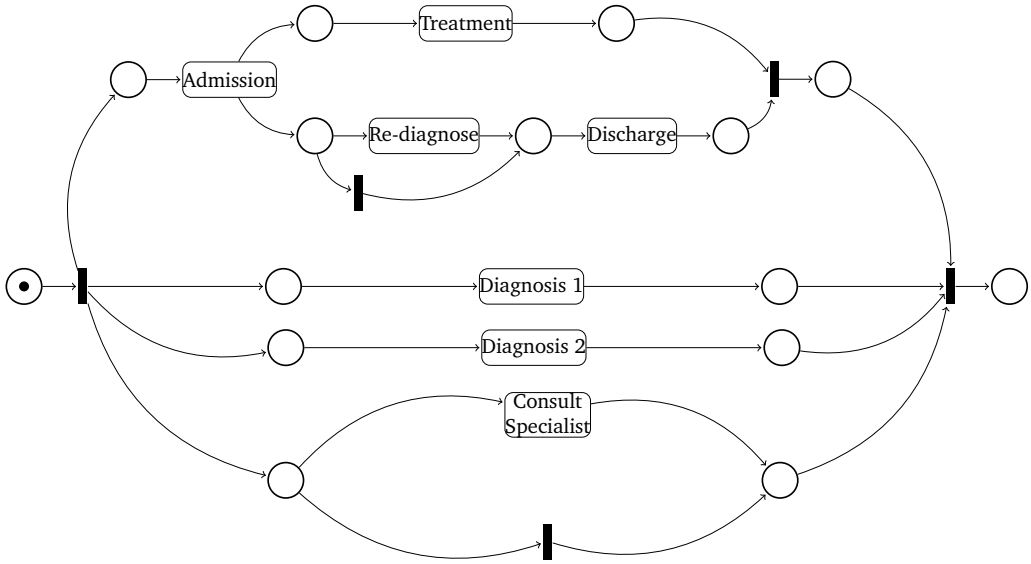


Figure 1.7: Process model discovered using inductive miner infrequent [110] using a noisy event log.

cision, etc. Typically, the automated discovery algorithms are *black-boxes* and hence, certain decisions made by the discovery technique may often be inaccurate and/or unclear to the end-user. Hence, even when the domain knowledge is missing/inadequate, it may still be advantageous to deduce process models interactively, by keeping the human-in-the-loop.

One way of incorporating user knowledge could be by *explicitly* specifying the domain knowledge, for example, via some domain-specific rules. The process mining techniques can then include the specified domain rules, in combination with the data from the event logs. Another way of incorporating user knowledge could be by *implicitly* including it in a human-in-the-loop setting. In such a scenario, the user would be involved in interactively discovering the results supported by process mining techniques. In such an interactive setting (i.e., implicitly considering user knowledge), the user has more expressibility, as the restrictions imposed by the language used for representing domain rules do not apply. However, it is also very important to guarantee certain runtime efficiencies. The argument of runtime efficiency could be relaxed when considering the user knowledge explicitly (non-interactive setting).

In this thesis, we aim to address this gap between process modeling and process mining, by devising techniques that enable incorporation of domain knowledge both

explicitly and implicitly. Majority of the techniques proposed in this thesis focus on implicit aspect of specifying domain knowledge, i.e., interactive techniques. For certain scenarios, wherein interactive techniques are not an optimal choice, we resort to developing techniques which include domain knowledge explicitly. In particular, we have identified four goals to address interactive process mining. In the upcoming sections, we discuss each of these goals in detail, and briefly describe our approach to address each goal.

1.4.1 Goal 1: Interactive Event Log Repair

The first goal of this thesis is to incorporate user knowledge in the source of process mining analysis, i.e., the event log. Data quality issues in the event log are not uncommon in a real-life setting. Event data may have different types of noise. Noise can be seen as (i) incorrectly logged, (ii) infrequent, or (iii) non-normative. An event log may contain noisy information due to multiple reasons, such as incorrect manual entry of events, incorrect processing of events, multiple data sources etc. Therefore, it is imperative to correct the information in the event log. One of the primary ways of *fixing* data quality issues could be based on user knowledge. For example, consider Case 3 from the event log of Table 1.1. It might be so that the second *Consult Specialist* event is *misplaced* and should instead be placed after the activity *Treatment*. A process/data analyst may be able to identify and repair the ordering of such incorrectly ordered events in the event log. Hence, the first goal of this thesis is to provide techniques to assist the user in interactively repairing event logs. A repaired event log can then be used by any automated process mining technique.

Our Approach

To support process mining analysis, we primarily focus on the control-flow aspect of the event data quality, i.e., (incorrect) ordering of events in the event log. In order to enable interactive repair of an event log, we identified three subgoals. We briefly discuss these subgoals along with the design choices made in our approach below.

- In order to repair an event log, it is first necessary to explore and detect the presence of possible ordering-related issues in the event log. As the user is interested in repairing the event logs, we believe it is ideal to *automatically* detect the possible issues that exist in the event log. Hence, in our approach, we use statistical approaches in combination with threshold-based filtering approaches, to automatically indicate to the user the presence of possible data quality issues in the event log.

- Once ordering-related issues in the event log have been identified, it is imperative to allow the user to correct (some) of these issues using the domain knowledge. This subgoal relates directly to the main goal of interactive event log repair. We believe it is ideal to correct multiple ordering-related issues at once, rather than tediously correcting issues on a case-by-case basis. Hence,

we provide a process-fragment-based approach to correct the ordering-related issues in the event log.

- Upon repairing the event log with the domain knowledge, the user may be interested in exploring the impact of change between the original event log and the repaired event log. In order to analyze the impact of the changes made in the event log after the repair, we provide a list of aggregated statistics to compare the original the repaired event log. The user could thus discard the changes, or keep the changes based on the impact that the changes had on the event log.

1.4.2 Goal 2: Interactive Process Discovery

The second research goal of this thesis aims at incorporating user knowledge in order to interactively discover process models. The aim here is to address the short-comings of traditional automated process discovery and hand-made process modeling.

Process discovery techniques aim to utilize the information from the event log in order to discover process models. However, they are constrained by the representation of the process language supported by the discovery technique. Most state-of-the-art process discovery techniques do not allow the possibility of incorporating a user in ordering and positioning of activities. Hence, the resulting process models may describe the event log extremely well, however may still be completely incomprehensible to the end user. Furthermore, most of the state-of-the-art process discovery techniques are incapable in discovering multiple occurrences of the same activity. Hence, most discovery techniques would not be able to correctly discover the process model from Figure 1.3, as the activity *Consult Specialist* occurs twice. Many discovery techniques also have difficulties in discovering constructs such as inclusive choices. Moreover, most automated discovery techniques do not allow specification of user knowledge during process discovery. Such domain knowledge may especially be handy when the event log contains some noisy and/or incomplete information.

The shortcomings of automated process mining techniques can be overcome by incorporating user knowledge during the process of process discovery. This points towards the top-down process modeling setting. Therefore, the aim here is to ensure that the user can have sufficient impact during the process modeling/discovery phase. It is imperative that the process being modeled is *valid*, i.e., sound, and does not result in *problems*, for example, deadlocks. Moreover, there needs to be an active support for modeling the process models by the user. This needs to be derived based on the information extracted from the event log. The feedback from the event logs, and the guarantees about soundness of process models, should be performed in a fast-enough way, which is suitable in an interactive setting. This is especially important, as it is undesirable to have long waiting times. In essence, the aim is to formulate interactive process modeling/discovery technique(s) where the *bottom-up* data-driven process discovery and *top-down* human-driven process modeling approaches “meet-in-the-middle”.

Our Approach

We addressed the goal of interactive process modeling/discovery using *sound free-choice workflow nets*, guided by the so-called *synthesis rules* for free-choice Petri nets [58]. The choice of using sound free-choice workflow nets and synthesis rules was based on the underlying subgoals of interactive process modeling/discovery. We briefly discuss these subgoals below, and subsequently motivate the design choices made for each of these subgoals.

- *Broad structural representations*: It is important that the class of process models discoverable in an interactive process modeling/discovery setting, supports basic structural constructs such as exclusive choices, concurrency and loops. Furthermore, it is also desirable that the class of process models supported allows for modeling of advanced operations such as inclusive choices and arbitrary (non-block structured) loops [167]. Process modeling languages such as process trees do not support construction of non-block structured processes. Hence, in order to support such broad structural representations, we use the class of free-choice workflow nets in our approach.

- *Soundness*: It is important that the process being modeled is sound. As mentioned already, soundness ensures important properties of the process such as absence of deadlocks, proper termination etc. BPMN models and/or arbitrary Petri net models can be unsound. In order to guarantee the soundness criteria, we use the liveness and boundedness preserving synthesis rules for free-choice Petri nets [58] in the context of free-choice workflow nets.

- *Incremental and Structured editing*: It is desirable to construct process models in an incremental and structured manner [43, 124]. With this as the goal, we support interactive editing of process models based on the synthesis rules, which inherently leads to an incremental and structured way of constructing the process model.

- *Event log support*: It is important that the user is provided with the information from the event log in order to perform data-driven decision making to construct process models. In order to address this, we take a leaf out of how some of the automated process discovery techniques perform decisions, by using abstracted information from the event log. We provide such abstracted information to the user in a tabular format, as well as project it on the process models, in order to support data-driven interactive process modeling.

- *Fast Conformance analysis*: For a process being modeled, the user would be interested in comparing it with the event log in order to analyze the conformance metrics such as the fitness and/or precision scores. The outcome from conformance checking could be used by the user as a means to discover better process models interactively. However, traditional conformance techniques could be too slow in the context of an interactive setting, and hence undesirable. Therefore, we use an abstraction-based technique to compute fast conformance analysis

to indicate the quality of the process model in comparison with the event log. Furthermore, we also use the underlying nature of the synthesis rules used to model the free-choice workflow nets in order to compute conformance for only those parts of the process model which could impact the conformance score, thereby further optimizing the performance.

- *Providing recommendations*: The user may need help during process modeling, especially when the positioning of an activity in the process model is unclear. That is, it is ideal to pre-compute and present some relevant process models for the user to choose from. Again, it is important that such recommendations are computed in an efficient way, suitable in an interactive setting. In order to address this, we use a pattern-based approach using the synthesis rules for free-choice workflow nets in order to pre-compute process models to be recommended.

1.4.3 Goal 3: Interactive Process Model Repair

The limitations of automated process discovery techniques typically also apply to automated process model repair techniques. Hence, it is imperative to include user knowledge about the process, together with a pre-existing process model and an event log, in order to repair a process model. For example, a domain expert would know that the activity Admission should be the first activity in the automatically discovered process model shown in Figure 1.7. The third goal addresses this concern of specifying a user knowledge, in order to interactively repair a process model.

Our Approach

The subgoals of interactive process modeling also apply to interactive process model repair. We address the goal of process model repair in two ways. First, we use an approach similar to the one used to address the goal of interactive process modeling/editing. Thereby, we also directly address all the subgoals of interactive process modeling/discovery, in the context of interactive process model repair. This approach has the advantage that it allows the user to intuitively consider domain knowledge while interactively repairing the process model.

There may be scenarios, wherein the user's domain knowledge about the process may be limited. Hence, the user may want to specify such domain knowledge explicitly first, and may seek for process models which describe both the specified domain knowledge and the information from the event log. It is preferable to show to the user multiple process models, each of which would have a different quality, so that the user can choose the most suitable and comprehensible model. Doing this in a completely interactive setting is far from trivial. Hence, we propose an approach which is based on a genetic technique, which takes the process model, the event log and the domain knowledge specified as constraints as its input, and returns (possibly many) process models, which may have different levels of quality, in terms of the number of constraints satisfied and the quality with respect to the event log. The user can

then interactively explore the different process models resulting from the technique, to select the most appropriate process model.

1.4.4 Goal 4: Interactive Process Analytics

The fourth goal of this thesis focuses on exploring information in the event log and the corresponding process model. The user should be able to explore what went on in the process, based on real-life execution as recorded by the event log and depicted by the process model. The goal is to come up with a toolkit that enables the user to interactively extract some of the common compliance and performance oriented questions in process mining, using process mining and visual analytics.

Our Approach

The fourth goal of this thesis primarily presents techniques for compliance, performance and data visualizations. In order to address this, we use state-of-the-art process mining and visualizations to extend the techniques proposed in the previous goals.

1.5 Thesis Roadmap

Figure 1.8 shows an overview of the different chapters from the thesis, with the aim of addressing the research goals discussed in the previous section. The theme of the thesis centers around interactive process mining, hence the user is central to all the four research goals as shown in Figure 1.8. In Chapter 2, we provide preliminaries relevant throughout the thesis. We begin the contents of the thesis with the foundational chapters, followed by the chapters related to the four research goals.

Chapter 3 and Chapter 4 present the foundations that are used to address the goals of the thesis. In Chapter 3, we discuss the building blocks that serve as the basis for enabling interactive process modeling, using the synthesis rules for free-choice nets from [58]. Primarily, this chapter discusses the applications of the synthesis rules from [58] in the context of free-choice workflow nets, and shows that these rules can indeed be applied to correctly deduce any possible free-choice workflow net structure.

Chapter 4 addresses the performance problems related to computing the building blocks as proposed in Chapter 3. A brute force approach is infeasible in practice. Therefore, an incremental approach is proposed to compute the building blocks in a fast-enough way, suitable in an interactive setting, while preserving the properties guaranteed in Chapter 3.

Chapter 8 focuses on addressing the first goal of this thesis. This chapter focuses on assisting the user in addressing data quality issues in the event log. Primarily, this chapter focuses on quality issues related to the ordering of events in the event log. Possible data quality issues are automatically detected, and the user can perform repairs by modeling process fragments, enabled by the foundations from Chapter 4. Furthermore, the user is also provided with the impact of the changes performed.

Chapter 5 focuses on addressing the second goal of this thesis. The foundations from Chapter 4 are combined with the information from the event log in order to

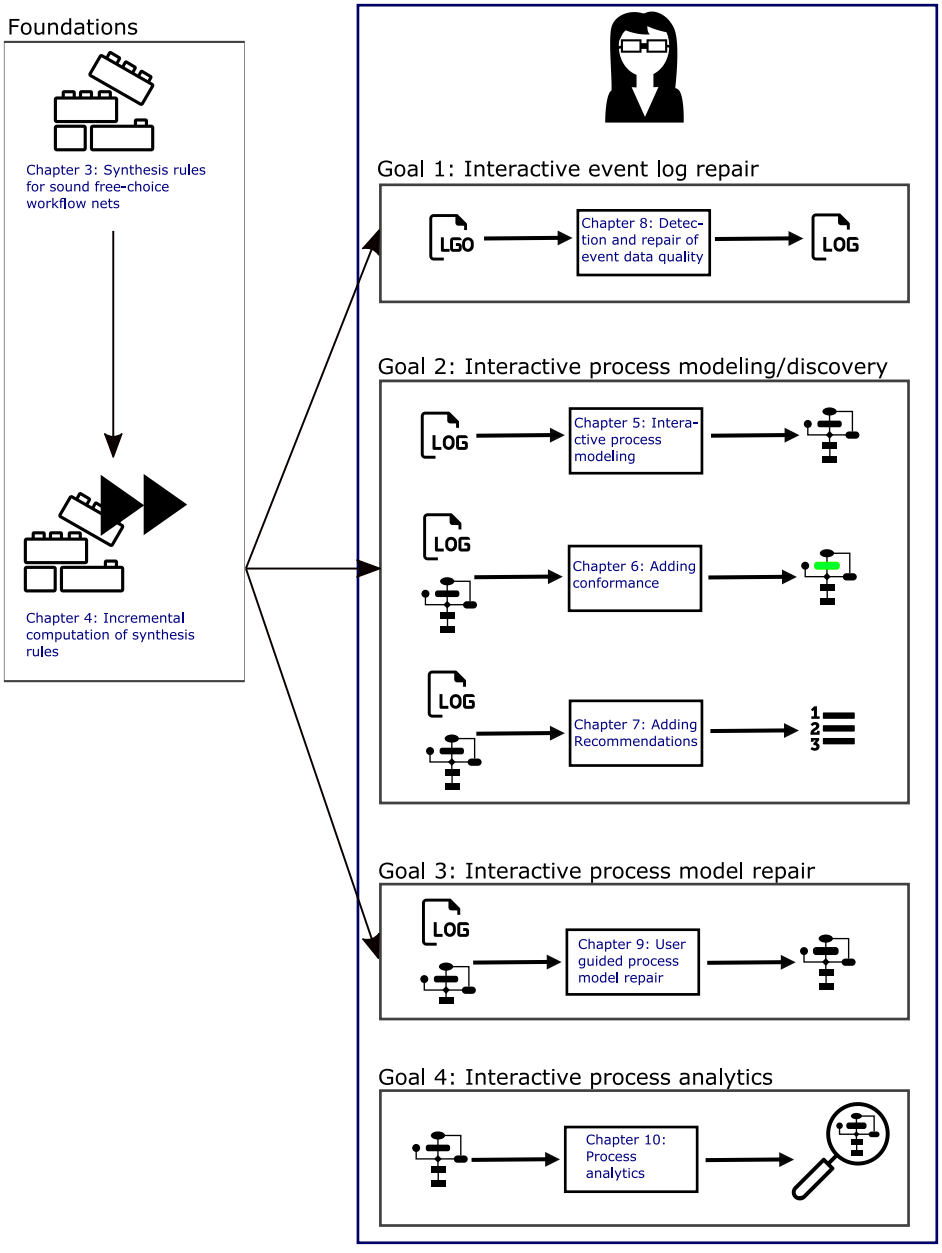


Figure 1.8: Outline of the thesis.

enable an interactive process modeling system. The user is provided with information extracted from the event log in order to make informed decisions while modeling a process model. Chapter 6 and Chapter 7 further improve the modeling/discovery of process models by calculating fast conformance and providing recommendations.

Chapter 6 provides techniques to compute approximated conformance scores between the process being modeled and the event log. State-of-the-art conformance techniques could be slow in certain circumstances. This is undesirable in an interactive setting, and hence the technique proposed in Chapter 6 trades accuracy for speed, and is therefore fast-enough, and hence more suitable in an interactive process modeling setting. Conformance analysis further assists the user in judging the impact of a change based on the information from the event log.

Chapter 7 moves further towards the automated process discovery setting, by automatically providing multiple recommendations to a user to discover a process model. The recommendations are ranked based on the conformance scores computed using Chapter 6. Moreover, the user can also switch between choosing from recommendations or over-riding all the recommendations and modeling process by hand based on the domain knowledge.

Chapter 9 focuses on addressing the third goal of the thesis, i.e., repairing process models by specifying domain knowledge. Two approaches are proposed. First, the foundations from Chapter 4 are used, along with some aspects from Chapter 5 in order to enable the user to interactively edit/repair a process model. Second, a technique is proposed to pre-specify some rules that should hold in the process model. The process model is then iteratively repaired based on the specified rules and the event log, and the user is provided with, possibly many, process models to choose from.

The fourth goal of this thesis is addressed in Chapter 10. Conformance technique [5] from the literature is used to compare a pre-existing process model and an event log. Furthermore, the system is enhanced with interactive capabilities such that the user can analyze the process model and evaluate data-attributes using visual analytic techniques.

The implementation details of the proposed solutions are discussed in the individual chapters. We wrap up by providing some conclusions, limitations of the proposed solutions and some of the future research directions in Chapter 11.

Chapter 2

Preliminaries

This chapter introduces some basic definitions and background of the concepts used throughout this book. We start with the introduction of bags and functions, followed by event logs, Petri nets, workflow nets and free-choice nets. The definitions corresponding to each type of net are sub-classified based on structural and behavioral properties.

2.1 Bags, Functions and Sequences

A bag (also called a multiset) is used to represent the state of a Petri net and to describe event logs, when a single trace is repeated multiple times. A bag over some set S is a function from S to the natural numbers that assigns only to a finite number of elements from S a positive value. For a bag B over set S and $s \in S$, $B(s)$ denotes the number of occurrences of s in B, often called the cardinality of s in B. Note that a finite *set* of elements of S is also a bag over S, namely the function yielding 1 for every element in the set and 0 otherwise. The set of all bags over set S is denoted $\mathscr{B}(S)$. We use brackets to explicitly enumerate a bag and superscripts to denote cardinalities. For example, $B = [a^2, b^3, c]$ denotes a bag B which contains the elements a, b and c 2, 3 and 1 times respectively. Bag B is a subbag of bag B', denoted $B \leq B'$, if and only if, for all $s \in S$, $B(s) \leq B'(s)$. The standard set operators can be extended to bags, for example, $a \in B$, $\{a\} \cup B = [a^3, b^3, c]$, $|B|=6$ etc.

A relation $R \subseteq X \times Y$ is a set of pairs, where $\pi_1(R) = \{x \mid (x, y) \in R\}$ is the domain of R, $\pi_2(R) = \{y \mid (x, y) \in R\}$ is the range of R, and $\omega(R) = \pi_1(R) \cup \pi_2(R)$ are the elements of R. For example, $\omega(\{(a, b), (b, c)\}) = \{a, b, c\}$. $f \in X \not\rightarrow Y$ is a partial function with domain $dom(f) \subseteq X$ and range $rng(f) = \{f(x) \mid x \in X\} \subseteq Y$. $f \in X \rightarrow Y$ is a total function, i.e., $dom(f) = X$.

Definition 1 (Function projection). *[162] Let $f \in X \not\rightarrow Y$ be a (partial) function and $Q \subseteq X$. $f \downarrow_Q$ is the function projected on Q: $dom(f \downarrow_Q) = dom(f) \cap Q$ and $f \downarrow_Q (x) = f(x)$ for $x \in dom(f \downarrow_Q)$.*

Such a projection could also be used for bags, example, $[a^2, b^3, c] \downarrow_{\{a,c\}} = [a^2, c]$.

$\sigma = \langle x_1, x_2, \cdots, x_n \rangle \in X^*$ denotes a sequence over X of length n. $\langle\rangle$ is the empty sequence. Sequences are used to represent paths in a graph and traces in an event log. $\sigma_1 \cdot \sigma_2$ is the concatenation of two sequences and $\sigma \downarrow_Q$ is the projection of σ on Q.

Definition 2 (Sequence projection). *[162] Let X be a set and $Q \subseteq X$ one of its subsets. $\downarrow_Q \in X^* \rightarrow Q^*$ is a projection function and is defined recursively: (1) $\langle\rangle \downarrow_Q = \langle\rangle$ and (2) for $\sigma \in X^*$ and $x \in X$:*

$$(\langle x \rangle \cdot \sigma) \downarrow_Q = \begin{cases} \sigma \downarrow_Q, & \text{if } x \notin Q; \\ \langle x \rangle \cdot \sigma \downarrow_Q, & \text{if } x \in Q \end{cases}$$

So $\langle a, b, a \rangle \downarrow_{\{a,d\}} = \langle a, a \rangle$. Functions can also be applied to sequences: if $dom(f) = \{a, d\}$ then $f(\langle a, b, a \rangle) = \langle f(a), f(a) \rangle$.

2.2 Event logs

Event logs record the actual process behavior, which is typically recorded in the information systems of an organization.

Definition 3 (Event log). *[163] Let $\mathscr{A}$ be the set of all process activities. An event e is the occurrence of an activity: $e \in \mathscr{A}$. A trace σ is a (possibly empty) sequence of events: $\sigma \in \mathscr{A}^*$. We denote an empty trace as $\langle\rangle$, and use $\langle \sigma_1, \ldots, \sigma_n \rangle$ to denote a non-empty trace σ, where n is the length of σ, that is, $n = |\sigma|$. Note that for every $1 \leq i \leq |\sigma|$, σ_i is an event. An event log L is a non-empty bag of traces, that is, $L \in \mathscr{B}(\mathscr{A}^*) \wedge L \neq []$.*

Note that the same trace may appear multiple times in an event log. Each trace corresponds to an execution of a process, i.e., a case. For example, consider an event log $L_1 = [< a, b, c >^5, < a, b, d >^5, < a, b, c, d >^{10}]$. L_1 contains information about 20 cases, for example a trace $< a, b, c >$ is a sequence followed by 5 cases.

2.3 Petri Nets

We use Petri nets as a means to represent process models.

2.3.1 Structure

In this subsection, we discuss the structural properties of Petri nets. We begin with the definition of Petri nets.

Definition 4 (Petri net). *A Petri net N is a tuple (P, T, F) such that:*

- *P is a finite set of places,*
- *T is a finite set of transitions such that $P \cap T = \emptyset$, and*

– $F \subseteq (P \times T) \cup (T \times P)$ *is the set of arcs, also called the flow relation.*

Figure 2.1 shows an example of a Petri net. $\{i, p_1, p_2, \cdots, o\}$ is a set of places and $\{\top, t_1, t_2, \cdots, \bot\}$ is a set of transitions respectively.

For any node $n \in P \cup T$, the *preset* of n in N, denoted ${}^{N}\bullet n$, is the set of all nodes that have an arc to n, that is, ${}^{N}\bullet n = \{n' | (n', n) \in F\}$. Likewise, for any node $n \in P \cup T$, the *postset* of n in N, denoted $n{}^{N}\bullet$, is the set of all nodes that have an arc from n, that is, $n{}^{N}\bullet = \{n' | (n, n') \in F\}$.

In case the net N is clear from the context, we typically omit it from these notations and use $\bullet n$ instead of ${}^{N}\bullet n$, etc. For example, in Figure 2.1, $\bullet p_3 = \{t_2, t_6, t_8\}$ and $t_5 \bullet = \{p_7, p_8\}$. We also extend the notions of preset and postset to sets of nodes, which results in sets of nodes. If $X \subseteq P \cup T$, then ${}^{N}\bullet X$ is defined as the set union of the respective presets, that is, ${}^{N}\bullet X = \bigcup_{n \in X} {}^{N}\bullet n$. Likewise, $X{}^{N}\bullet = \bigcup_{n \in X} n{}^{N}\bullet$. We would like to emphasize that in our use of notation we slightly deviate from the conventional setting, such that we consider the preset (or postset) of a set of nodes results in a *set* of nodes as the output, instead of a *bag* of nodes.

Definition 5 (Strongly connected Petri net)**.** *A Petri net $N = (P, T, F)$ is called* strongly connected *if and only if $\forall_{x,y \in P \cup T}(x, y) \in F^*$, where F^* is the reflexive transitive closure of F.*

Figure 2.1 can be turned into a strongly connected Petri net by removing the place o and re-routing the outgoing arc from $\bot$ to place i as shown in Figure 2.2a.

Definition 6 (Net projection)**.** *Let $N = (P, T, F)$ be a Petri net, let $P' \subseteq P$ be a subset of places, and let $T' \subseteq T$ be a subset of transitions. The* projection *of net N on $P' \subseteq P$ and $T' \subseteq T$, denoted $N \downarrow_{P'}^{T'}$ is defined as the net $N' = (P', T', F')$ where $F' = F \cap ((P' \times T') \cup (T' \times P'))$.*

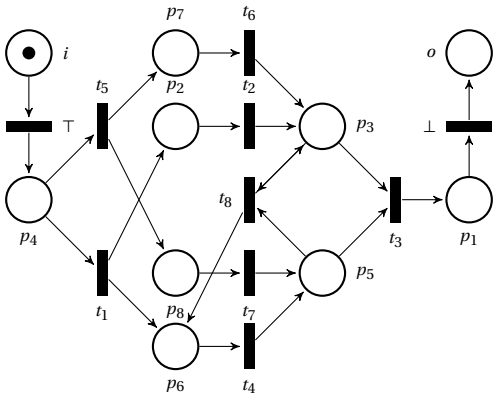


Figure 2.1: An example Petri net.

Figure 2.2b shows the net from Figure 2.2a projected on places $\{i, p_4, p_2, p_7, p_3, p_1\}$ and transitions $\{\top, t_1, t_5, t_2, t_6, t_3, t_8, \bot\}$.

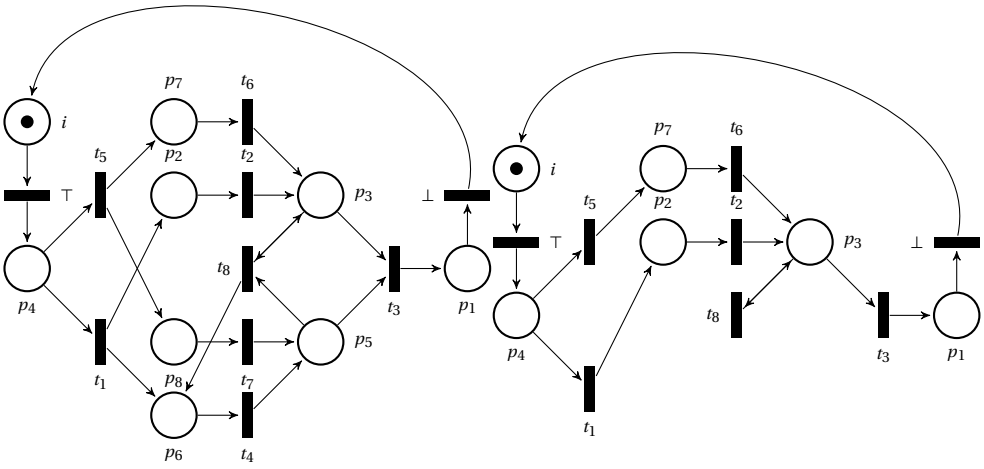
Definition 7 (S-net, S-component, S-coverability)**.** *Let $N = (P, T, F)$ and $N' = (P', T', F')$ be two Petri nets such that $N' = N\downarrow_{P'}^{T'}$. The net N' is called an* S-net *if and only if for all $t' \in T'$ it holds that $|{}^{N'}\bullet t'| = 1 = |t'^{N'}\bullet|$, that is, every transition has a single place in its preset and a single place in its postset. The S-net N' is called an* S-component *of net N if and only if N' is strongly connected and for all $p' \in P'$ it holds that ${}^{N}\bullet p' \cup p'^{N}\bullet \subseteq T'$. The net N is called* S-coverable *if and only if for every place $p \in P$ there exists an S-component that contains p.*

Figure 2.2b is also an example of an S-net. Furthermore, the net from Figure 2.2b is an S-component of the net from Figure 2.2a.

Definition 8 ((Proper) Siphons and Traps)**.** *Let $N = (P, T, F)$ be a Petri net. A siphon is defined as a set of places $S \subseteq P$, such that: $\bullet S \subseteq S\bullet$. If S is non-empty, then it is a proper siphon. A trap is defined as a set of places $S \subseteq P$, such that: $S\bullet \subseteq \bullet S$. If S is non-empty, then it is a proper trap.*

In Figure 2.2b, the set of places $\{i, p_1, p_2, p_3, p_4\}$ forms a proper siphon, whereas the set of places $\{i, p_1, p_2, p_3, p_4, p_7\}$ forms a proper trap.

Definition 9 (Incidence matrix)**.** *Let $N = (P, T, F)$ be a Petri net. The incidence matrix*



(a) An example strongly connected net. (b) An example projected net.

Figure 2.2: Strongly connected net and projected net, S-net and S-component.

Table 2.1: The incidence matrix of the net from Figure 2.2a.

	$\top$	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	$\bot$
i	-1	0	0	0	0	0	0	0	0	1
p_4	1	-1	0	0	0	-1	0	0	0	0
p_2	0	1	-1	0	0	0	0	0	0	0
p_3	0	0	1	-1	0	0	1	0	0	0
p_1	0	0	0	1	0	0	0	0	0	-1
p_5	0	0	0	-1	1	0	0	1	-1	0
p_6	0	1	0	0	-1	0	0	0	1	0
p_7	0	0	0	0	0	1	-1	0	0	0
p_8	0	0	0	0	0	1	0	-1	0	0

of N, denoted $\mathbf{N}$*, is the matrix* $\mathbf{N} : (P \times T) \to \{1, 0, -1\}$ *such that*

$$\mathbf{N}(p,t) = \begin{cases} -1, & \text{if } (p,t) \in F \text{ and } (t,p) \notin F; \\ 1, & \text{if } (p,t) \notin F \text{ and } (t,p) \in F; \\ 0, & \text{otherwise.} \end{cases}$$

The column vector $\mathbf{t} : P \to \{-1, 0, 1\}$ *of* $\mathbf{N}$ *is associated to a transition t in the net N. Similarly, the* row vector $\mathbf{p} : T \to \{-1, 0, 1\}$ *of* $\mathbf{N}$ *is associated to a place p in the net N.*

Table 2.1 shows the incidence matrix of the net from Figure 2.2a.

2.3.2 Behavior

Having discussed the structural properties of Petri nets, we now discuss some behavioral properties.

Definition 10 (Marking)**.** *Let $N = (P, T, F)$ be a Petri net. A marking M of N is a bag over the places of N, that is, $M \in \mathcal{B}(P)$.*

A marking M is represented as a collection of $M(p)$ *tokens* for every place p. Removing a token from p then corresponds to removing one occurrence of p from M, and adding a token to p corresponds to adding one occurrence of p to M. The marking of the Petri net from Figure 2.2a is $[i]$ i.e., a single token in place i.

Definition 11 (Transition enabledness, firing)**.** *Let $N = (P, T, F)$ be a Petri net, let M be a marking of N, and let $t \in T$ be a transition of N. Transition t is* enabled *in M, denoted $M \xrightarrow{t}$, if and only if $\bullet t \leq M$. An enabled transition may* fire*, which leads to a new marking M_1, where $M_1 = (M - \bullet t) + t\bullet$, that is, in the new marking, a token is first removed from every place in the preset of t and a token is then added to every place in the postset of t. This firing is denoted $M \xrightarrow{t} M_1$.*

In Figure 2.2a, transition $\top$ is enabled, as all the incoming places, i.e. i, is marked. Hence, transition $\top$ can be fired, which results in the marking $[p_4]$.

Corollary 1 (Number of tokens does not change in an S-component). *Let $N = (P, T, F)$ be a Petri net, and let $N' = (P', T', F')$ be an S-component of N. The accumulated number of tokens in P' does not change when firing any transition from N in any marking.*

Proof. This is straightforward to check, as any transition in T' removes one token from P' and adds one token to P', whereas any transition in $T \setminus T'$ does not touch P'. □

Definition 12 (Occurrence sequences, Reachable marking). *Let $N = (P, T, F)$ be a Petri net and let M be a marking of N. If $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \cdots \xrightarrow{t_n} M_n$ are transition occurrences, then $\sigma = \langle t_1, t_2, \cdots, t_n \rangle$ is an occurrence sequence leading from the marking M to M_n, and is denoted by $M \xrightarrow{\sigma} M_n$.*

This also includes an empty sequence $\langle\rangle$, $M \xrightarrow{\langle\rangle} M$ for every marking M.

A marking M' is called reachable from marking M, denoted $M \xrightarrow{\sigma} M'$, if and only if there exists an occurrence sequence σ, such that $M \xrightarrow{\sigma} M'$.

We use $R(N, M)$ to denote the set of markings reachable from marking M in Petri net N: $R(N, M) = \{M' | M' \in \mathscr{B}(P) \wedge M \xrightarrow{\sigma} M'\}$. The marking $[p_7, p_8] \in R(N, [i])$ in the net N of Figure 2.2a.

Definition 13 (Liveness, Deadlock-freedom, Boundedness, Safeness). *Let $N = (P, T, F)$ be a Petri net, and let M be a marking of N. A transition $t \in T$ is called* live *in M if and only if for all $M' \in R(N, M)$ there exists an $M'' \in R(N, M')$ such that $M'' \xrightarrow{t}$, that is, transition t can get enabled from any marking reachable from M. The net N is called* live *in M if and only if all its transitions are live in M. The net (P, T, F) is* deadlock-free *in M if and only if for all $M' \in R(N, M)$ there exists $t \in T$, such that $M' \xrightarrow{t}$, that is, every reachable marking enables at least one transition. A place $p \in P$ is called* k-bounded *in M, where k is a natural number, if and only if for all $M' \in R(N, M)$ it holds that $M'(p) \leq k$, that is, place p never holds more than k tokens. A place $p \in P$ is called* bounded *in M if and only if there exists a natural number k such that p is k-bounded in M. A place $p \in P$ is called* unbounded *in M if and only if it is not bounded in M. A place $p \in P$ is called* safe *in M if and only if it is 1-bounded in M. The net N is called* bounded *in M if and only if all its places are bounded in M, it is called* unbounded *in M otherwise. The net N is called* safe *in M if and only if all its places are safe in M.*

The Petri net from Figure 2.2a is live, bounded, deadlock-free and safe in $[i]$.

2.4 Workflow Nets

Having discussed Petri nets, we now discuss workflow nets, which provide a way to represent Petri nets in terms of process models.

2.4.1 Structure

As was the case with Petri nets, we discuss workflow net in terms of their structural properties first, followed by their behavioral properties. We begin with the definition of workflow net.

Definition 14 (Workflow net). *A workflow net W is a tuple $(P, T, F, i, o, \top, \bot)$ such that:*

- *(P, T, F) is a Petri net,*
- *$i \in P \wedge \bullet i = \emptyset$, that is, i is a place with an empty preset,*
- *$o \in P \wedge o\bullet = \emptyset$, that is, o is a place with an empty postset,*
- *$\top \in T \wedge i\bullet = \{\top\} \wedge \bullet\top = \{i\}$,*
- *$\bot \in T \wedge \bot\bullet = \{o\} \wedge \bullet o = \{\bot\}$, and*
- *$\forall_{n \in P \cup T} (i, n) \in F^* \wedge (n, o) \in F^*$, that is, every node n is on some path from i to o.*

Figure 2.3 shows the generic structure of a workflow net. Note that we build upon the definition of workflow nets from [158], by including $\top$ and $\bot$ transitions. A workflow net $(P, T, F, i, o, \top, \bot)$ is also a Petri net (P, T, F). For this reason, we allow ourselves to use a workflow net where a Petri net is expected. Figure 2.1 shows an example of a workflow net.

Definition 15 (Short-circuited workflow net). *Let $W = (P, T, F, i, o, \top, \bot)$ be a workflow net. The short-circuited net of W, denoted by $SC(W)$, is obtained by removing the place o, and adding an arc from $\bot$ to i. The set of places of a short-circuited workflow net, denoted as ${}^{SC}P$, is $P \setminus \{o\}$. Therefore, $SC(W)$ is the Petri net $({}^{SC}P, T, (F \setminus \{(\bot, o)\}) \cup \{(\bot, i)\})$.*

Figure 2.2a shows an example of short-circuiting a workflow net from Figure 2.1.

2.4.2 Behavior

In this section, some behavioral properties of workflow nets are discussed.

Definition 16 (Initial, final, and reachable markings [158]). *Let $W = (P, T, F, i, o, \top, \bot)$ be a workflow net. The initial marking of W is the marking $[i]$, the final marking is the marking $[o]$, and the reachable markings of W are all markings that are reachable from the initial marking, that is, $R(W, [i])$.*

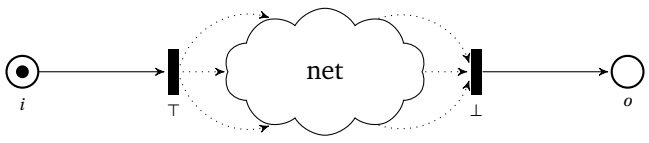


Figure 2.3: An example workflow net structure. Place i is the only source place and place o is the only sink place. $\top$ and $\bot$ are always the first and last transitions resp.

As the initial marking is implicit in a workflow net, we typically write $R(W)$ instead of $R(W,[i])$.

Definition 17 (Soundness [168]). *Let $W = (P,T,F,i,o,\top,\bot)$ be a workflow net. W is called* sound *if the following three conditions hold:*

1. *For every marking $M \in R(W)$ it holds that $M \xrightarrow{\sigma} [o]$, that is, from every marking reachable from the initial marking, the final marking can be reached.*
2. *For every marking $M \in R(W)$ such that $[o] \leq M$ it holds that $[o] = M$, that is, the final marking is the only reachable marking that includes place o.*
3. *For every transition $t \in T$ there is a marking $M \in R(W)$ such that $M \xrightarrow{t}$, that is, every transition can be fired in some reachable marking.*

The workflow net from Figure 2.1 is a sound workflow net. The soundness property of a workflow net can be related to the liveness and boundedness property of Petri nets as shown in Theorem 1.

Theorem 1 (Soundness vs. liveness and boundedness [158]). *Let $W = (P,T,F,i,o,\top,\bot)$ be a workflow net. W is sound if and only if the net $SC(W)$ is live and bounded in $[i]$.*

Proof. See [158]. □

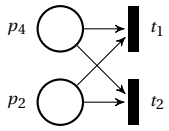
2.5 Free-choice Nets

We now discuss a special class of Petri nets, called free-choice nets.

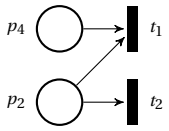
2.5.1 Structure

Definition 18 (Free-choice net). *[58] Let $N = (P,T,F)$ be a Petri net. Petri net N is called a* free-choice net *if and only if for every two places $p, p' \in P$ either $p\bullet \cap p'\bullet = \emptyset$ or $p\bullet = p'\bullet$.*

Historically, we can make a distinction between strong and weak free-choice nets [58]. A strong free-choice net implies that for every place p having an arc to a transition t either:



(a) A free-choice net construct.



(b) A non-free-choice net construct.

Figure 2.4: Example of a free-choice net and a non-free-choice net construct.

- t is the only output transition of p (which implies that t cannot be in conflict with any other transition), or
- p is the only input place of t (which implies that there is no synchronization at t).

Figure 2.4 shows a weak free-choice net, also known as extended free-choice net. Since we only consider the weaker condition in our setting, we do not distinguish between strong and weak free-choice nets.

2.5.2 Behavior

In this subsection, we first define a behavioral property of Petri nets called well-formedness. Then we use this behavioral property, to demonstrate the relationship between well-formedness and soundness in the context of free-choice workflow nets.

Definition 19 (Well-formedness). *[58] Let $N = (P, T, F)$ be a free-choice net. The net N is called* well-formed *if and only if a marking $M \in \mathscr{B}(P)$ exists such that N is live and bounded in M.*

Corollary 2 (Sound workflow nets correspond to well-formed nets). *Let $W = (P, T, F, i, o, \top, \bot)$ be a sound workflow net. Then the Petri net $SC(W)$ is well-formed.*

Proof. As W is a sound workflow net, we know that the Petri net $SC(W)$ is live and bounded in $[i]$. □

Theorem 2 (Well-formedness induces S-coverability). *[58] Let $N = (P, T, F)$ be a well-formed free-choice net. Then the net N is S-coverable.*

Proof. See [58]. □

Theorem 3 (Liveness and deadlock-freedom). *Let $W = (P, T, F, i, o, \top, \bot)$ be a free-choice workflow net and let $SC(W)$ be bounded. Then $SC(W)$ is live in $[i]$ if and only if it is deadlock-free in $[i]$.*

Proof. Follows from Theorem 4.31 in [58] and the fact that a short-circuited free-choice workflow net is strongly connected. □

Corollary 3 (Sound free-choice workflow nets are safe). *Let $W = (P, T, F, i, o, \top, \bot)$ be a sound free-choice workflow net. Then the Petri nets $SC(W)$ and (P, T, F) are safe in marking $[i]$ (from Theorem 3 in [159]).*

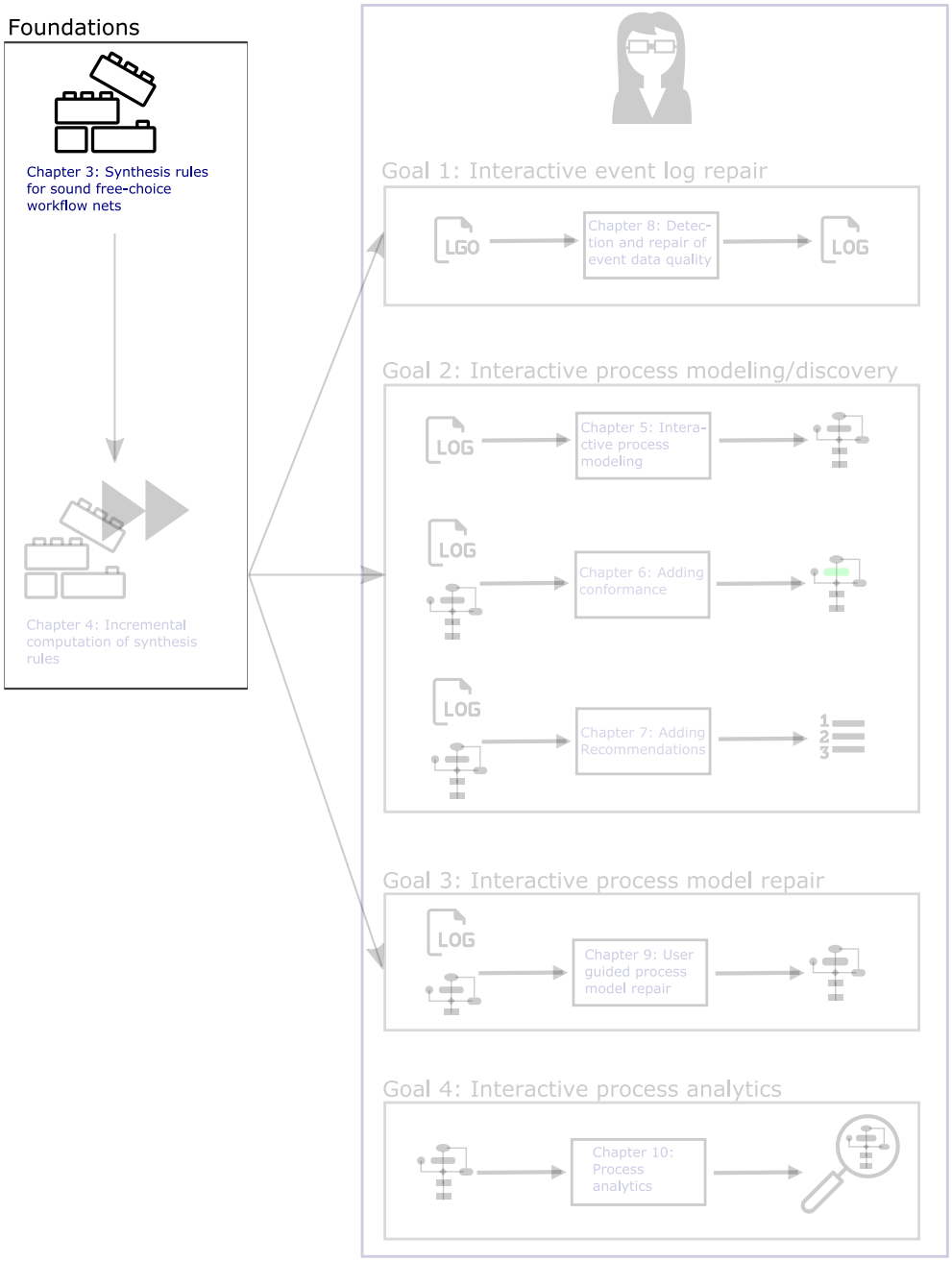
Proof. As W is a sound workflow net, we know that the Petri net $SC(W)$ is well-formed, and is hence S-coverable. As the marking contains only a single token in place i, and as every place p is covered by some S-component, place p cannot contain more than one token in $R(SC(W), [i])$. Note that $[o] \in R((P, T, F), [i])$ but $[o] \in R(SC(W), [i])$. Using this, we can say that $R((P, T, F), [i]) \subseteq R(SC(W), [i])$. Therefore, place p cannot contain more than one token in $R((P, T, F), [i])$. □

2.6 Summary

In this chapter, we presented the concepts and definitions which form the basis for the upcoming chapters. We first introduced bags and functions, followed by the discussion of event logs, which are central to process mining analysis. We then discussed different classes of Petri nets and their properties. We particularly focused on free-choice workflow nets, as this is the class of Petri nets that is used in the upcoming chapters to represent process models.

Chapter 3

Synthesis Rules for Sound Free-choice Workflow nets


Foundations
Chapter 3: Synthesis rules for sound free-choice workflow nets
Chapter 4: Incremental computation of synthesis rules
Goal 1: Interactive event log repair
LOG
Chapter 8: Detection and repair of event data quality
LOG
Goal 2: Interactive process modeling/discovery
LOG
Chapter 5: Interactive process modeling
LOG
Chapter 6: Adding conformance
LOG
Chapter 7: Adding Recommendations
1
2
3
Goal 3: Interactive process model repair
LOG
Chapter 9: User guided process model repair
Goal 4: Interactive process analytics
Chapter 10: Process analytics

In this chapter, we focus on discussing and proposing an engine that enables interactive process modeling. The modeling notation used to represent a process model is critical in any process modeling/discovery setting. Many modeling notations are possible, for example, process trees, BPMN, UML, Petri nets. Ideally, the modeling notation should support common process modeling constructs, such as sequences, exclusive choices, concurrency and loops [167]. In our case, we are also interested in complex structural representations such as inclusive choices and non-block structuredness that allows for constructs such as arbitrary loops. Notations such as process tree are limited, i.e., cannot be used for representing non-block structured process models. A work-around for this could be to duplicate activities, as shown in the block-structured Petri net equivalent in Figure 3.1. However, this results in additional nodes (p_{10} and t_{13}) in the process model. Thus the complexity of the process model increases, thereby deteriorating the end result [124].

Most of the high-level process modeling notations can be translated into Petri nets. Moreover, the behavioral semantics enabled by Petri nets provide a robust way

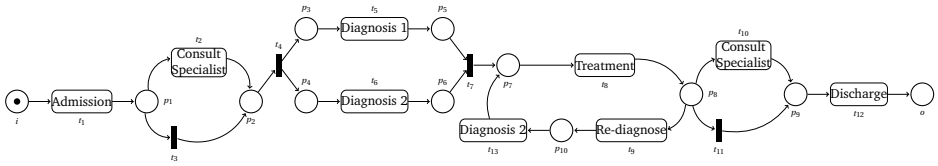


Figure 3.1: Converting the non-block structured care-pathway process into a block structured care-pathway process by duplicating the activity Diagnosis 2 (transitions t_6 and t_{13}).

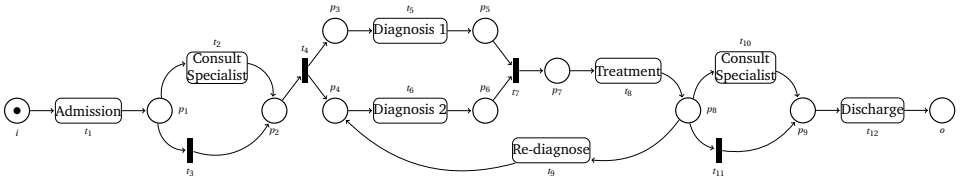


Figure 3.2: An unsound process model, obtained by removing the arc from t_9 to p_5 in Figure 1.3.

in performing process mining analysis. Hence, in our approach we focus on a class of Petri nets, called workflow nets which support both the simple and complex constructs discussed above. The care-pathway process model (Figure 1.3) shows an example of a workflow net.

Allowing the user to model any process model could result in an invalid (unsound) process model. That is, the process being modeled should satisfy the soundness criteria, to ensure important properties such as the absence of deadlocks in the process model, proper termination of the process etc. Ideally, a user should only be allowed to perform operations that result in valid process models. Consider Figure 3.2, which shows an example of an unsound process model obtained by removing the arc from t_9 to p_5 in Figure 1.3. This process model is a workflow net, but it is unsound as t_7 cannot be fired after executing t_9 and t_6, due to a missing token at p_5. In order to avoid such a scenario, we have an additional requirement that the user should only be allowed to model valid process models. That is, we should have some building blocks which only lead to valid (sound) process models, as shown in Figure 3.3.

Process trees result only in sound process models, and hence can serve as such building blocks. However, as already discussed, a major limitation of using process trees is the difficulty in representing non-block-structured process models. Therefore, in our approach, we use the so-called synthesis rules from [58] as the building blocks of sound free-choice workflow nets. Since the synthesis rules were originally defined for a class of Petri nets called free-choice nets, we focus on a sub-class of workflow nets called free-choice workflow nets. The synthesis rules for free-choice nets are known to preserve liveness and boundedness. We extend these rules in the context of free-choice workflow nets, and make use of the liveness and boundedness properties to show that the extended synthesis rules preserve soundness in free-choice workflow nets.

In Section 3.1, we discuss the problem definition, and the context and requirements placed on these building blocks. In Section 3.2, we provide a solution which uses a synthesis-rules-based approach for constructing free-choice workflow nets. In

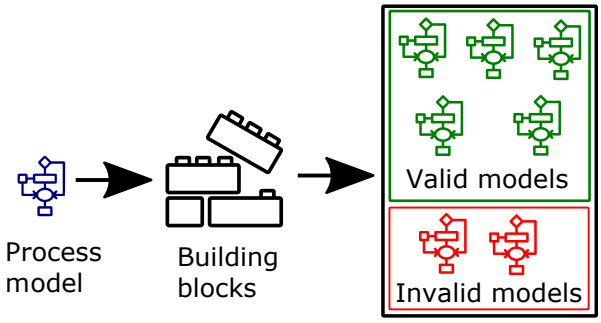


Figure 3.3: Building blocks are used to generate process models from a starting process model. Ideally, the building blocks should be able to automatically identify and allow only *valid* process models.

Section 3.3.1, we show that the presented approach is correct, and in Section 3.3.2 we show that the proposed approach is complete. Next, we discuss the performance evaluation in Section 3.4. We discuss the related work in Section 3.5, followed by conclusions in Section 3.6.

3.1 Problem Definition: Need of Building Blocks for Sound Workflow Nets

In order to enable interactive process modeling, we need a process editing engine as a starting point. As discussed in Goal 2 in Chapter 1, this editing engine should support several subgoals. Among these, the first couple of subgoals include support for complex structural representations, such as non-block structured constructs and inclusive choices, and guarantees of soundness, i.e., the modeled process model should always be sound. As discussed previously, free-choice workflow nets in combination with the synthesis rules could be used to address these sub-goals. Therefore, in our approach we would like to consider free-choice workflow nets to represent process models, and use the synthesis rules as the building blocks for these process models. With this in mind, we aim to address the following research question in this chapter:

- How can we build a robust editing engine that provides building blocks for enabling interactive editing of free-choice workflow nets, such that
  - the building blocks can be defined for free-choice workflow nets. Originally, the synthesis rules were defined in the context of free-choice nets [58]. As we are only interested in free-choice workflow nets, the first challenge is to port the original synthesis rules into the context of free-choice workflow nets.
  - only valid free-choice workflow nets can be generated. We know that free-choice workflow nets support complex constructs such as non-block structured process models. However, it is also important that the free-choice workflow nets are valid in nature, i.e., the building blocks should result only in sound free-choice workflow nets. Therefore, it is important to show that the extended synthesis rules in the context of free-choice workflow nets result only in sound free-choice workflow nets.
  - any valid free-choice workflow net can be constructed. That is, starting with an initial *empty* free-choice workflow net, we should be able to construct any sound free-choice workflow net using the editing engine based on the building blocks of the synthesis rules.

In Section 3.2, we discuss how we can make use of the synthesis rules as the building blocks for sound free-choice workflow nets. In Section 3.3, we discuss the correctness and completeness of these rules in the context of sound free-choice workflow nets.

3.2 Synthesis Rules as Building Blocks

Synthesis rules are a set of rules that allow synthesizing larger Petri nets using smaller Petri nets. A synthesis rule kit contains multiple rules, such that each rule results in a different operation on a given Petri net. A synthesis rules kit, such as the one provided in [58] provide guarantees on the generation of any well-formed free-choice net. The well-formedness property of these nets can also be related to the soundness property of workflow nets. Hence, such a synthesis rule kit forms an ideal solution to the question posed in Section 3.1. In Section 3.2.1 we discuss the original synthesis rules kit from [58], and in Section 3.2.2 we put these rules in the context of sound free-choice workflow nets.

3.2.1 Synthesis Rules for Well-formed Free-choice nets

In this section, we discuss the synthesis rules from [58], as they form the skeleton of our approach. The synthesis rule kit from [58] contains three rules: (i) abstraction rule, (ii) linearly dependent place rule, and (iii) linearly dependent transition rule. It has been proven that these rules are complete and can be used to synthesize any well-formed free-choice net. The synthesis rules described in [58] are valid for *all* well-formed free-choice nets. The initial net for these synthesis rules is a live and bounded atomic net containing only one place and transition as shown in Figure 3.4a.

The abstraction rule allows expansion of a net by adding a new place and a new transition. The abstraction rule can be formally defined as:

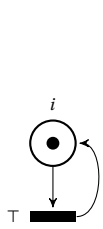
Definition 20 (Abstraction rule ψ_A (derived from [58]))**.** *Let $N = (P, T, F)$ and $N' = (P', T', F')$ be two free-choice nets. N' is synthesized from N, i.e., $(N, N') \in \psi_A$ if and only if:*

- $P' = P \cup \{p\}$ *(where $p \notin P$)*
- $T' = T \cup \{t\}$ *(where $t \notin T$)*
- $R \times S \subseteq F$ *such that $R \subseteq T$, $S \subseteq P$ and $R \times S \neq \emptyset$*
- $F' = (F \setminus (R \times S)) \cup ((R \times \{p\}) \cup (\{p\} \times \{t\}) \cup (\{t\} \times S))$

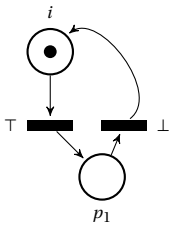
The free-choice net from Figure 3.4b is synthesized from the initial free-choice net of Figure 3.4a, by adding a place p_1 and a transition $\perp$ using the ψ_A rule ($R = \{\top\}, S = \{i\}$). Similarly, the free-choice net from Figure 3.4c is synthesized from the free-choice net of Figure 3.4b, by adding a place p_3 and a transition t_3 using the ψ_A rule ($R = \{\top\}, S = \{p_1\}$). The opposite of the ψ_A synthesis rule is called the ϕ_A reduction rule, which removes a place and a transition from the net.

Next, we discuss the two linearly dependent synthesis rules. A linearly dependent rule allows for the introduction of a new linearly dependent place or a linearly dependent transition in a free-choice net. We begin by first introducing linearly dependent transitions and places, followed by the definitions of the rules.

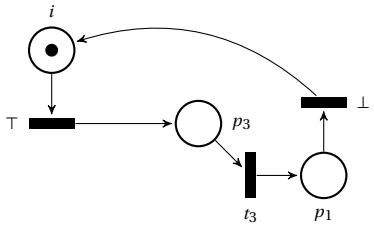
From basic linear algebra we know that a column vector **c** belonging to a matrix **M** is linearly dependent if and only if **c** can be expressed as some linear combination



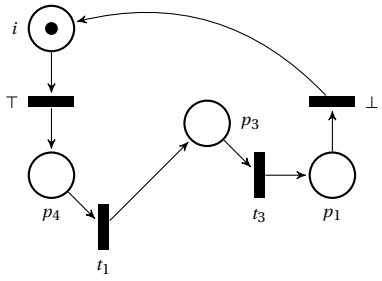
(a) Initial atomic free-choice net net.



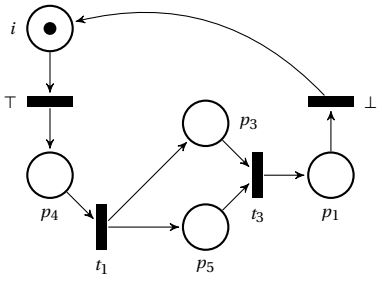
(b) Adding $\perp$ and p_1 using ψ_A.



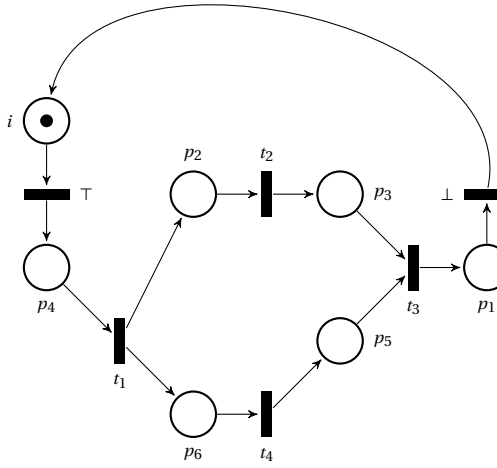
(c) Adding t_3 and p_3 using ψ_A.



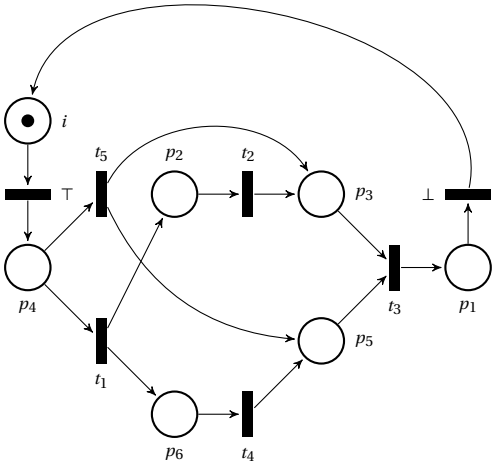
(d) Adding t_1 and p_4 using ψ_A.



(e) Adding p_5 using ψ_P.



(f) Adding t_2, p_2 and t_4,p_6 using ψ_A.



(g) Adding t_5 using ψ_T.

Figure 3.4: Example usage of synthesis rules starting with initial atomic net.

	$\top$	t_1	t_3	$\bot$
i	-1	0	0	1
p_1	0	0	1	-1
p_3	0	1	-1	0
p_4	1	-1	0	0
p_5	0	1	-1	0

(a) Incidence matrix corresponding to Figure 3.4e.

	$\top$	t_1	t_2	t_3	t_4	$\bot$	t_5
i	-1	0	0	0	0	1	0
p_1	0	0	0	1	0	-1	0
p_2	0	1	-1	0	0	0	0
p_3	0	0	1	-1	0	0	1
p_4	1	-1	0	0	0	0	-1
p_5	0	0	0	-1	1	0	1
p_6	0	1	0	0	-1	0	0

(b) Incidence matrix corresponding to Figure 3.4g.

Figure 3.5: Incidence matrices after usage of linear dependency rules.

of the other columns in **M**. Using this, we define linearly dependent nodes as follows:

Definition 21 (Linearly dependent nodes (derived from [58]))**.** *Let $N = (P, T, F)$ be a Petri net and* **N** *be the incidence matrix of N. Let $\mathbb{Q}$ be the set of all rational numbers.*
A place p of N is linearly dependent if there exists a vector $\lambda : P \to \mathbb{Q}$ such that $\lambda(p) = 0$ and $\lambda \cdot \mathbf{N} = \mathbf{p}$.
Similarly, a transition t of N is linearly dependent if there exists a vector $\mu : T \to \mathbb{Q}$ such that $\mu(t) = 0$ and $\mathbf{N} \cdot \mu = \mathbf{t}$.

Following the definition of linearly dependent nodes, we now discuss the two linear dependency rules. Figure 3.5a shows an example of linearly dependent row vector (which is a part of the incidence matrix).

Definition 22 (Linear dependent transition rule ψ_T(derived from [58]))**.** *Let $N = (P, T, F)$ and $N' = (P', T', F')$ be two free-choice nets. N' is synthesized from N, i.e., $(N, N') \in \psi_T$ if and only if:*

- $P' = P$
- $T' = T \cup \{t\}$, *where* $t \notin T$
- $F' = F \cup \tilde{F}$, *where* $\tilde{F} \subseteq ((P \times \{t\}) \cup (\{t\} \times P))$
- *t is a linearly dependent transition of N'*
- $\overset{N'}{\bullet} t \cup t \overset{N'}{\bullet} \neq \emptyset$

Definition 23 (Linear dependent place rule ψ_P (derived from [58]))**.** *Let $N = (P, T, F)$ and $N' = (P', T', F')$ be two free-choice nets. N' is synthesized from N, i.e., $(N, N') \in \psi_P$ if and only if:*

- $T' = T$
- $P' = P \cup \{p\}$, *where* $p \notin P$

– $F' = F \cup \tilde{F}$, *where* $\tilde{F} \subseteq ((\{p\} \times T) \cup (T \times \{p\}))$

– p *is a linearly dependent place of* N'

– $^{N'}\bullet p \cup p^{N'}\bullet \neq \emptyset$

The free-choice net from Figure 3.4e is synthesized from the free-choice net of Figure 3.4d, by adding a linearly dependent place p_5 using the ψ_P rule. From Figure 3.5a, it is easy to see that the row vector corresponding to place p_5 is the same as the row vector corresponding to place p_3, i.e., $\mathbf{p_5} = \mathbf{p_3}$, and hence p_5 is indeed a linearly dependent place. In Figure 3.4e, this implies that p_5 and p_3 have a similar effect on the free-choice net.

The free-choice net from Figure 3.4g is synthesized from the free-choice net of Figure 3.4f, by adding a linearly dependent transition t_5 using the ψ_T rule. From Figure 3.5b, we can see that the column vector corresponding to t_5 has the same overall effect as the column vectors corresponding to transitions t_1, t_2 and t_4. That is, $\mathbf{t_5} = \mathbf{t_1} + \mathbf{t_2} + \mathbf{t_4}$.

The opposite of linearly dependent synthesis rules ψ_T and ψ_P are the linearly dependent reduction rules ϕ_T and ϕ_P, which remove a linearly dependent transition or a place from the net.

It has been shown that along with ϕ_A, these reduction rules are complete for well-formed free-choice nets [58]. That is, any well-formed free-choice net can be reduced to an atomic net containing only one place and one transition.

Theorem 4 (Synthesis rules preserve well-formedness [58])**.** *Let* $N = (P, T, F)$ *and* $N' = (P', T', F')$ *be two free-choice nets such that* $(N, N') \in \psi_A \cup \psi_P \cup \psi_T$*. Then* N' *is well-formed if and only if* N *is well-formed.*

3.2.2 From Free-choice Nets to Sound Free-choice Workflow Nets

The synthesis rules from [58] are defined in the context of well-formed free-choice nets. However, from a process mining perspective, we are interested in process models and hence workflow nets. Therefore, in this section, we define the new rules based on the synthesis rules from [58] for valid free-choice workflow nets. Before we begin with the new definitions of synthesis rules, we define the initial sound free-choice workflow net.

Definition 24 (Initial sound free-choice workflow net W_0)**.** *The initial sound free-choice workflow net is* $W_0 = (\{i, p_1, o\}, \{\top, \bot\}, \{(i, \top), (\top, p_1), (p_1, \bot), (\bot, o)\}, i, o, \top, \bot)$.

The starting net W_0 for our approach is shown in Figure 3.6a. Place i is the only source and initially marked place in the net, and place o is the only sink place in the net. Figure 3.6b is the short-circuited version of initial free-choice workflow net. Having defined the initial free-choice workflow net, we now describe the corresponding synthesis rules with respect to sound free-choice workflow nets.

Definition 25 (Abstraction rule ψ_A^{WF} for free-choice workflow nets)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ and $W' = (P', T', F', i, o, \top, \bot)$ be two free-choice workflow nets. W' can be synthesized from W, denoted $(W, W') \in \psi_A^{WF}$, if and only if $(SC(W), SC(W')) \in \psi_A$.*

Note that the abstraction rule allows for the addition of a new place and a new transition, between a (set of) transition(s) and place(s). However, it is not possible to use the abstraction rule between the transition $\bot$ and the sink place o, as the resulting net would no longer be a workflow net (Definition 14).

Definition 26 (Linear dependency in sound free-choice workflow nets)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ be a sound free-choice workflow net. Let $\mathbb{Q}$ be the set of all rational numbers. Let ${}^{SC}P = P \setminus \{o\}$. A place $p \in {}^{SC}P$, is called linearly dependent in W if and only if there exists a vector $\lambda : {}^{SC}P \to \mathbb{Q}$ such that $\lambda(p) = 0$ and $\lambda \cdot SC(\mathbf{W}) = \mathbf{p}$.*

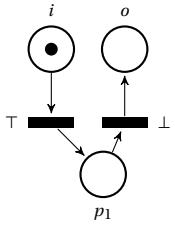
A transition $t \in T$ is called linearly dependent in W if and only if there exists a vector $\mu : T \to \mathbb{Q}$ such that $\mu(t) = 0$ and $SC(\mathbf{W}) \cdot \mu = \mathbf{t}$.

It should be noted that for row vectors, we consider a vector mapping to ${}^{SC}P$ which is a $|P| - 1$-dimensional vector. This is due to the fact that in a short-circuited version of a free-choice workflow net, place o is removed. Using Definition 26, we define the linearly dependent place and transition rules.

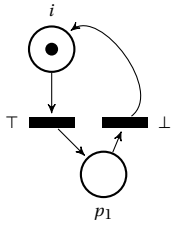
Definition 27 (Linearly dependent place rule ψ_P^{WF} for free-choice workflow nets)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ and $W' = (P', T', F', i, o, \top, \bot)$ be two free-choice workflow nets. W' can be synthesized from W using ψ_P^{WF}, denoted $(W, W') \in \psi_P^{WF}$, if and only if:*

1. *$(SC(W), SC(W')) \in \psi_P$ and*
2. *All proper siphons in $SC(W')$ contain i*

The second requirement in Definition 27 is required to limit synthesis to only *sound* free-choice workflow nets, by avoiding linearly dependent places which would result in deadlocks. The need for this condition is elaborated later in Theorem 7.



(a) Initial free-choice workflow net.



(b) Short-circuited version of initial free-choice workflow net.

Figure 3.6: Initial workflow net.

Definition 28 (Linearly dependent transition rule ψ_T^{WF} for free-choice workflow nets)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ and $W' = (P', T', F', i, o, \top, \bot)$ be two free-choice workflow nets. W' can be synthesized using ψ_T^{WF}, denoted $(W, W') \in \psi_T^{WF}$, if and only if $(SC(W), SC(W')) \in \psi_T$.*

It should be noted that every free-choice net, other than Figure 3.4a, from Figure 3.4 can be seen as short-circuited versions of free-choice workflow nets. Therefore, starting with Figure 3.4b as the initial short-circuited free-choice workflow net, we can use the synthesis rules for free-choice workflow nets and synthesize the free-choice workflow nets shown in Figure 3.4.

3.3 Synthesis Space

Having described the synthesis rules corresponding to free-choice workflow nets, we now discuss how these can be used to enable interactive process modeling. The concept of *synthesis space* is central to enable interactive process editing. Loosely speaking, the synthesis space contains all the possible applications of ψ_A^{WF} , ψ_P^{WF} or ψ_T^{WF} corresponding to a free-choice workflow net.

We know that a free-choice workflow net can be extended by adding one place and/or transition at a time using the synthesis rules. In an interactive setting, a user would typically name such a place and/or transition. In order to replicate such a scenario, we first define an abstract function which takes a free-choice workflow net as an input and returns a fresh place and/or a fresh transition as output.

Definition 29 (Place and transition identifier)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ be a free-choice workflow net. Then, $\mathbb{I}_{\mathbb{P}}(W)$ denotes an abstract function that returns a fresh place $p \notin P$. Similarly, $\mathbb{I}_{\mathbb{T}}(W)$ denotes an abstract function that returns a fresh transition $t \notin T$.*

Using Definition 29, we now define the synthesis space as follows:

Definition 30 (Synthesis Space SS)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ be a sound free-choice workflow net, and let F_{WN} be the universe of sound free-choice workflow nets. Consider a fresh place $p = \mathbb{I}_{\mathbb{P}}(W)$ and a fresh transition $t = \mathbb{I}_{\mathbb{T}}(W)$. The synthesis space $SS(W) = SS_A(W) \cup SS_P(W) \cup SS_T(W)$, where:*

- $SS_A(W) = \{W' = (P', T', F', i, o, \top, \bot) \in F_{WN} \mid (W, W') \in \psi_A^{WF} \wedge \{p\} = P' \setminus P \wedge \{t\} = T' \setminus T\}$,
- $SS_T(W) = \{W' = (P', T', F', i, o, \top, \bot) \in F_{WN} \mid (W, W') \in \psi_T^{WF} \wedge P' = P \wedge \{t\} = T' \setminus T\}$,
- $SS_P(W) = \{W' = (P', T', F', i, o, \top, \bot) \in F_{WN} \mid (W, W') \in \psi_P^{WF} \wedge T' = T \wedge \{p\} = P' \setminus P\}$

In essence, the newly added node(s) can be identified by any name as depicted by the abstract function in Definition 29. We mainly use the abstract function to keep the synthesis space finite. Therefore, the synthesis space contains a finite number of

ways in which a place $\mathbb{I}_{\mathbb{P}}(W)$ and/or a transition $\mathbb{I}_{\mathbb{T}}(W)$ can be added to a free-choice workflow net W using one of the synthesis rule. That is, $SS_A(W)$, $SS_P(W)$ and $SS_T(W)$ are all disjoint for a free-choice workflow net W. This is evident from the fact that $SS(W)$ always synthesizes a net W by adding a transition $\mathbb{I}_{\mathbb{T}}(W)$ and/or a place $\mathbb{I}_{\mathbb{P}}(W)$. For example, using ψ_A^{WF} leads to adding a new transition $\mathbb{I}_{\mathbb{T}}(W)$ *and* a new place to $\mathbb{I}_{\mathbb{P}}(W)$ to W, whereas $SS_P(W)$ and $SS_T(W)$ synthesizes the net by adding a new place $\mathbb{I}_{\mathbb{P}}(W)$ (using ψ_P^{WF}) *or* a new transition $\mathbb{I}_{\mathbb{T}}(W)$ (using ψ_T^{WF}). It should be noted that the synthesis space essentially caters to the change in structure of a free-choice workflow net upon usage of a synthesis rule. Conceptually, the newly added place $\mathbb{I}_{\mathbb{P}}(W)$ and/or transition $\mathbb{I}_{\mathbb{T}}(W)$ can then be named as desired by the user. For example, consider the free-choice workflow net of Figure 3.4c, which is in the synthesis space of Figure 3.4b. In this net, the newly added nodes are named as p_3 and t_3 resp.

Figure 3.7 shows an overview of the usage of the synthesis space to enable interactive process editing. Starting with a sound free-choice workflow net W_0, any free-choice workflow net can be selected from the initial synthesis space $SS(W_0)$. Note that the nodes would always be named as desired. After a free-choice workflow net W' is chosen, the synthesis space $SS(W')$ corresponding to W' is recomputed. This is the 'Recompute Synthesis Space $SS(W')$' phase from Figure 3.7. The re-computation of synthesis space is done in a brute force way.

In the next section, we show that the synthesis rules for free-choice workflow nets preserve soundness. Furthermore, we show that, starting with an initial atomic free-choice workflow net, we can always deduce any sound free-choice workflow net using the synthesis rules (i.e., selecting nets from subsequent synthesis spaces and naming the nodes in the desired way).

3.3.1 Correctness of Synthesis Rules for Sound Free-choice Workflow Nets

In this section, we discuss the implications of using the synthesis rules in order to enable interactive process discovery. We first discuss the correctness of each derived

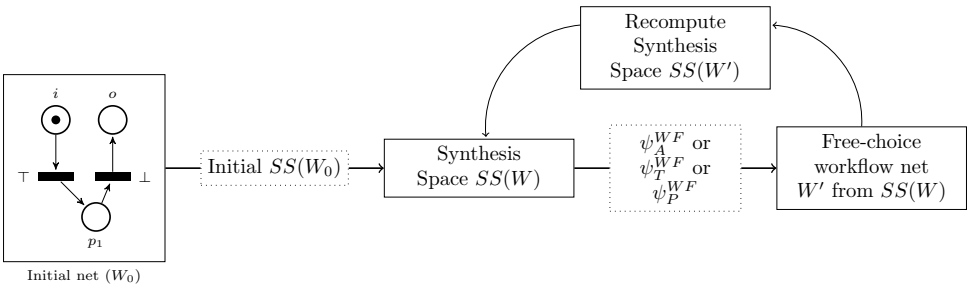


Figure 3.7: Synthesis space contains all the free-choice workflow nets which can be synthesized from a particular free-choice workflow net by using either ψ_A^{WF}, ψ_P^{WF} or ψ_T^{WF} rule. The synthesis space for the initial net is pre-computed.

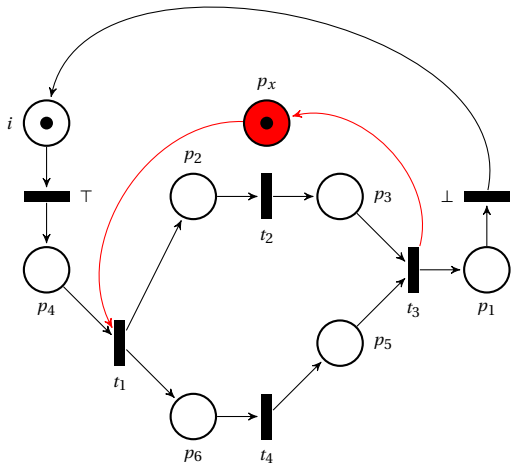
rule. As the new rules are defined in the context of free-choice workflow nets, we know that the net derived using any rule is also a free-choice workflow net. However, we also need to ensure that the derived net is also a sound free-choice workflow net, when the original net is a sound free-choice workflow net. Therefore, we show, on a case by case basis, that the application of each rule to a sound free-choice workflow net also results in sound free-choice workflow nets, starting with the abstraction rule (Definition 25) in Theorem 5.

Theorem 5 (ψ_A^{WF} preserves soundness)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ and $W' = (P', T', F', i, o, \top, \bot)$ be two free-choice workflow nets such that $(W, W') \in \psi_A^{WF}$. Then the net W' is sound if the net W is sound.*

Proof. To show that W' is sound, we need to show that $SC(W')$ is live and bounded in $[i]$ (Theorem 1).

Boundedness The net W is sound, hence $SC(W)$ is live and bounded in $[i]$, hence $SC(W)$ is well-formed. As $(SC(W), SC(W')) \in \psi_A$, $SC(W')$ is also well-formed, thus making it S-coverable, and hence bounded for any marking M', and hence bounded in $[i]$.

Liveness As $SC(W)$ is bounded in $[i]$ and strongly connected, it suffices to show that it contains no deadlocks in $[i]$, which follows from the facts that (1) $SC(W)$ does



(a) Adding p_x results in proper siphons $\{p_x, p_6, p_5\}$ and $\{p_x, p_2, p_3\}$, which do not contain the initial place i.

	$\top$	t_1	t_2	t_3	t_4	$\bot$
i	-1	0	0	0	0	1
p_1	0	0	0	1	0	-1
p_2	0	1	-1	0	0	0
p_3	0	0	1	-1	0	0
p_4	1	-1	0	0	0	0
p_5	0	0	0	-1	1	0
p_6	0	1	0	0	-1	0
p_x	0	-1	0	1	0	0

(b) Place p_x is linearly dependent as $\mathbf{p_x} = \mathbf{p_4} + \mathbf{p_1} + \mathbf{i}$.

Figure 3.8: Linearly dependent place that results in well-formed free-choice net, but not in a sound free-choice workflow net.

not contain deadlocks in $[i]$ and (2) by construction this rule cannot introduce new deadlocks, as the original rule (ψ_A) used to derive (ψ_A^{WF}) cannot introduce deadlocks [58].

□

Similar to ψ_A^{WF}, the linear dependency transition rule for free-choice workflow nets also *builds* upon the linear dependency transition rules defined for free-choice nets. Hence, in Theorem 6 we use a similar approach to show that the resulting net remains a sound free-choice workflow net after application of ψ_T^{WF} rule (Definition 28) on a sound free-choice workflow net.

Theorem 6 (ψ_T^{WF} preserves soundness)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ and $W' = (P', T', F', i, o, \top, \bot)$ be two free-choice workflow nets such that $(W, W') \in \psi_T^{WF}$. Then W' is sound if the net W is sound.*

Proof. To show that W' is sound, we need to show that $SC(W')$ is live and bounded in $[i]$.

Boundedness The net W is sound, hence $SC(W)$ is live and bounded in $[i]$, hence $SC(W)$ is well-formed. Hence, $SC(W')$ is also well-formed, hence S-coverable, hence bounded for any marking M', and hence bounded in $[i]$.

Liveness As $SC(W')$ is bounded in $[i]$ and strongly connected, it suffices to show that it contains no deadlocks in $[i]$, which follows from the facts that (1) $SC(W)$ contains no deadlocks and (2) it is clear that removing a transition t' cannot violate a pre-existing siphon. Hence, the introduction of a new transition t' cannot result in any additional siphons in the net W'. Since free-choice workflow nets an unmarked proper siphon without i results in deadlocks (Theorem 7), $SC(W')$ is deadlock free. Assume that $SC(W')$ is not live in $[i]$. Then from [58] (Theorem 5.8), we know that there is an S-component that is not marked in $[i]$. Hence, from [58] (Theorem 5.2), we know that there is a proper siphon that is not marked in $[i]$, that is, that does not contain i. As this is a contradiction according to the second condition of Definition 27, $SC(W')$ has to be live.

□

Unlike ψ_A^{WF} and ψ_T^{WF} which are rather straightforward extensions of ψ_A and ψ_T, in ψ_P^{WF} (Definition 27) there is an additional requirement of having the initially marked place i in all the proper siphons of the new net. We first demonstrate the need for such a condition using Figure 3.8. Adding a new place p_x in the net from Figure 3.8 may result in deadlock in the workflow net version of the short-circuited free-choice workflow net. Clearly, the newly added place is a linearly dependent place, and would fit the description of ψ_P. It should be noted that, after adding this place, the net is still a well-formed free-choice net. An example of an initial marking for which the net is live and bounded is $[i, p_x]$. However, in a free-choice workflow net the initial marking is always $[i]$. The second requirement from Definition 27 ensures that there are no unmarked siphons in the net, and hence no deadlocks. In order to prove this formally, we first provide Theorem 7.

Theorem 7 (A free-choice workflow net with an unmarked proper siphon without i contains dead transitions)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ be a sound free-choice workflow net. Let p be a linearly dependent place that is added to W, resulting in the net $W' = (P', T', F', i, o, \top, \bot)$, (such that $p \notin P$). If p results in a proper siphon S in $SC(W')$ such that $p \in S \wedge i \notin S$, then $SC(W')$ has dead transitions and hence W' is unsound.*

Proof. We know that $[i]$ is the initial marking of W. Since the newly added place $p \neq i$, it is unmarked. As the introduction of p results in a siphon S in $SC(W')$, by the definition of siphon we know that all transitions requiring tokens from S are dead transitions in $SC(W')$. As $SC(W')$ is strongly connected, such transitions exist. Hence $SC(W')$ has deadlocks and hence W' is unsound. □

Using Theorem 7, we discuss a corollary, which forbids the introduction of linearly dependent places having the same input and output transition sets.

Corollary 4. *A new place p containing the same inputs and outputs cannot be added to a sound free-choice workflow net using ψ_P^{WF}.*

Proof. If p has the same input and output transitions, then we have $\bullet p = p \bullet$. Therefore, p forms a set which is a proper siphon $\bullet p \subseteq p \bullet$. As the siphon $\{p\}$ does not contain the initially marked place i, adding p would result in an unsound free-choice workflow net. □

Finally, using Theorem 7 and Definition 27 we discuss how ψ_P^{WF} rule preserves soundness.

Theorem 8 (ψ_P^{WF} preserves soundness)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ and $W' = (P', T', F', i, o, \top, \bot)$ be two free-choice workflow nets such that $(W, W') \in \psi_P^{WF}$. Then the net W' is sound if the net W is sound.*

Proof. To show that W' is sound, we need to show that $SC(W')$ is live and bounded in $[i]$.

Boundedness The net W is sound, hence $SC(W)$ is live and bounded in $[i]$, hence $SC(W)$ is well-formed. Therefore, $SC(W')$ is also well-formed, hence S-coverable, hence bounded for any marking M', and hence bounded in $[i]$.

Liveness As $SC(W')$ is bounded in $[i]$ and strongly connected, it suffices to show that it contains no deadlocks in $[i]$. The second condition of Definition 27 forbids the introduction of unmarked proper siphons in the net. Assume that $SC(W')$ is not live in $[i]$. Then from [58] (Thm 5.8), we know that there is an S-component that is not marked in $[i]$. Hence, from [58] (Thm 5.2), we know that there is a proper siphon that is not marked in $[i]$, that is, that does not contain i. As this is a contradiction according to the second condition of Definition 27, $SC(W')$ has to be live.

□

After having shown that the synthesis rules for free-choice workflow nets preserve soundness, we now show that it is possible to deduce any sound free-choice workflow net starting with the initial free-choice workflow net.

3.3.2 Completeness of Synthesis Rules for Sound Free-choice Workflow Nets

In this section, we show that the synthesis rules for free-choice workflow nets are complete to synthesize any sound free-choice workflow net. From [58] (pg. 161), we know that any well-formed free-choice net can be synthesized using the synthesis rules from Subsection 3.2.1, starting with an initial atomic free-choice net.

Theorem 9. *Any sound free-choice workflow net W' can be derived from the initial net W_0 (Figure 3.6a) using the three synthesis rules ψ_A^{WF}, ψ_P^{WF} and ψ_T^{WF}.*

Proof. In order to show the completeness of the proof, we use the reduction rules [58] (reverse of synthesis rules) repeatably to obtain the initial atomic net. That is, we argue that we can always reduce a short-circuited free-choice workflow net that is live and bounded in $[i]$ to the initial short-circuited free-choice workflow net, and finally reduce it by using the abstraction rule in reverse on the place i to obtain the initial atomic free-choice net. We make use of the fact that the reduction rules (reverse of synthesis rules) for well-formed free-choice nets are complete, i.e., any well-formed free-choice net can be reduced to the initial atomic free-choice net by using the original synthesis rules for free-choice nets in reverse [58].

The proof uses Theorem 6.17 of [58], which states that a free-choice net (N, M) is live and bounded if, and only if (i) N is well-formed and (ii) M marks every proper siphon of N. Assume we have a free-choice workflow net that is live and bounded in $[i]$. As the short-circuited free-choice workflow net is well-formed and live and bounded in $[i]$, we know (Theorem 6.17 of [58]) that every proper siphon marks the place i. We now need to show that after having applied a possible synthesis rule 'in reverse' every siphon still marks place i, until it is no longer possible to apply any reduction rule (synthesis rule in reverse).

- ψ_A^{WF} : For the abstraction rule this is trivial. Using the abstraction rule in reverse would remove a place (and a transition) from the net. The siphons that included the removed place are now siphons with that place removed, and siphons that did not include the removed place are still siphons. Hence using abstraction rule in reverse cannot introduce new siphons in the net.

- ψ_P^{WF} : For the linearly dependent place rule, this is also trivial, as removing a place can only invalidate siphons, but not introduce them.

- ψ_T^{WF} : For the linearly dependent transition rule, this is more involved, as this rule could potentially introduce siphons. Assume that this rule has removed a transition such that a new minimal siphon was introduced. Note that every siphon includes a minimal siphon, so we may assume a minimal siphon here.

As removing the transition introduced the siphon, this transition adds tokens to the siphon, but does not remove tokens from it. From Prop. 5.4 of [58], this new minimal siphon is also a trap. As adding a transition that does not remove tokens from a trap cannot invalidate the trap, the minimal siphon is also a trap in the original short-circuited free-choice workflow net. As that net was live and bounded in $[i]$, this trap includes the place i. Hence, the introduced minimal siphon contains the place i.

Therefore, we can always apply some synthesis rule 'in reverse' on a short-circuited free-choice workflow net that is live and bounded in $[i]$, and the observation that applying the abstraction rule in reverse on the place i and transitions $\top$ and $\bot$ can be postponed until the end. Hence, we can synthesize any sound free-choice workflow net by synthesizing the corresponding short-circuited free-choice workflow net that is live and bounded in $[i]$. □

3.4 Evaluation

From Section 3.3.1, we know that the proposed approach correctly synthesizes only sound free-choice workflow nets. Furthermore, from Section 3.3.2, we know that it is possible to synthesize any sound free-choice workflow net starting with the initial atomic free-choice workflow net. In order to check the applicability of the approach in an interactive setting, we evaluate the time needed to calculate the synthesis space.

First, we examine the time taken to compute the synthesis space based on the size of a free-choice workflow net. Consider the net from Figure 3.8a, without the place p_x. There are 6 transitions ($\top, t_1, t_2, t_3, t_4$ and $\bot$) in the net. In order to calculate all the possible linearly dependent places that could be added to the net, we first have to compute the vectors based on all the permutations of the six transitions (with the values $-1, 0$ and 1), resulting in 120 vectors. Next, we have to compute if each of these vectors is linearly dependent on the rows of the incidence matrix of Figure 3.8b. This computation is non-trivial. A similar argument holds for computing the linearly de-

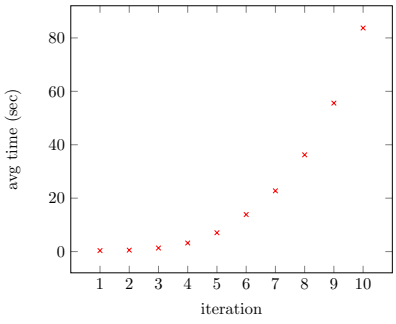


Figure 3.9: Performance of brute force approach for generating synthesis space after synthesizing 10 nets.

pendent transitions using the brute-force approach. As the size of the net grows, the time taken to compute the linear dependency rules in such a way grows exponentially due to the combinatorial blow-up. In order to provide an intuition, we use an empirical evaluation to compute all the applications of synthesis rules using the brute-force approach. We use a simple (non-parallelized) algorithm for computing the synthesis space on a machine with 2.70 GHZ processor with 64-bit OS and 8 GB of RAM.

Starting with the initial atomic free-choice workflow net W_0, a net W_1 can be synthesized by using any random synthesis rule. Then another random synthesis rule can be applied on the synthesized net W_1, to obtain a new net W_2. This can be repeated until the application of 10 synthesis rules. This experiment can be repeated any number of times. Let us assume we apply 10 synthesis rules in sequence, leading from $W_0, W_1, W_2, \cdots, W_{10}$. We repeated this experiment 100 times, to synthesize 1000 nets in total. The average of values for the time taken to compute the synthesis space using the brute force approach at each synthesis iteration was recorded and are plotted in Figure 3.9. As evident, the time taken for computing the synthesis space grows exponentially with the brute-force approach. This is clearly not acceptable in an interactive setting, where the response times should be minimal [127]. In the next chapter, we show how we can improve on this.

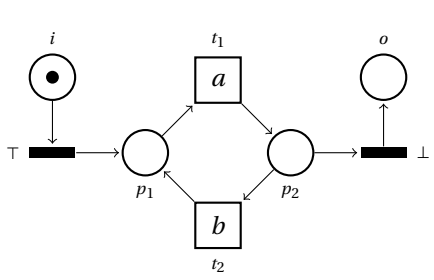
3.5 Related Work

In this section, we discuss the approaches from the literature similar to our approach. In particular, we discuss other synthesis kits which could have been used as building blocks for creating the interactive process modeling engine. Synthesis rules are basically reverse applications of the so-called *reduction rules*. The idea of reduction rules is to start with a bigger net, and iteratively use one of the rules from a reduction rule kit. Reduction rules are well-researched in literature, with [18,19,57,83] among the early contributors. Primarily, reduction techniques were used for verification of presence of certain properties in the nets, depending on how much a net could be reduced using a reduction rules kit, for example, [172,179]. An alternative to the usage of building blocks, and hence to synthesis rules, could have been an approach based on post-verification. That is, allowing any action to change a given workflow net, and checking the soundness of the modeled workflow net a posteriori using for example the reduction rules. [76] provides verification of several industrial business process models within milliseconds using [156,176,179,192]. However, such a solution in an interactive setting is not ideal, as it could first allow synthesis of unsound workflow nets, which would later be classified as unsound. However, by using synthesis rules as the building blocks, we guarantee that we are always within the realm of sound workflow nets.

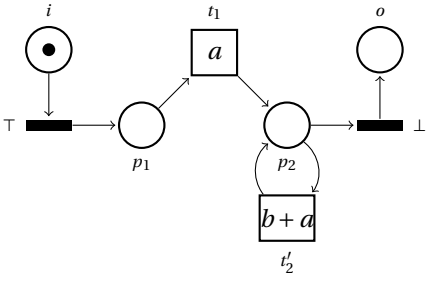
In [72], authors propose a reduction rule kit, which is correct and complete to reduce any colored free-choice workflow net. These have later been extended to the probabilistic case in [75]. In theory, these rules can be used in reverse, in order to synthesize any sound free-choice workflow net, starting with an atomic net. Moreover, unlike the rules discussed in this chapter which are devised from [58], all the

rules from [72] are *local*, i.e., it is not necessary to analyze the entire net to check the applicability of a rule. Therefore, these rules could be computed faster. However, we argue that these rules are not directly suitable for use in an interactive setting. We demonstrate this intuitively with an example using Figure 3.10. As shown, the reduction rules from [72] can be used to reduce the net from Figure 3.10a to the net of Figure 3.10c. In order to synthesize a net, the user has to proceed in the opposite direction. That is, in order to add b to the net from Figure 3.10c to derive the net from Figure 3.10a, the user first has to create an intermediate net by adding a self-loop transition, to derive the net from Figure 3.10b, and then use another rule to convert the self-loop to a loop. Hence, the rules from [72] are very involved, especially when it comes to the usage of the so-called *shortcut rule*. The user has to model intermediate structures in order to derive the desired nets, which can be tedious, especially when a user is interactively constructing a process model. Therefore, we argue that the rules derived based on [58] that are discussed in this chapter are straight-forward and hence are easier to understand for the user, making them suitable in an interactive setting. Using the ψ_T^{WF} rule, the user can directly add b to the net from Figure 3.10c to derive the net of Figure 3.10a, without the need of any intermediate net. Therefore, even though the rules from [72] can be computed in a faster way, they are not very intuitive and hence unsuitable for applications in an interactive setting.

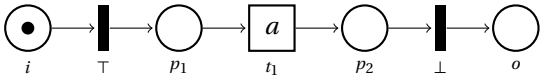
The technique of synthesizing large Petri nets from smaller Petri nets is not new and has been well-researched for over two decades [11, 49, 71, 74, 152]. In general, Petri net synthesis techniques have found tremendous applications in certain domains, such as automated manufacturing systems [37, 39, 102] and embedded software [101, 146]. Our main focus is not to suggest new synthesis rules, but to develop



(a) An example free-choice workflow net to demonstrate the applications of reduction rules from [72].



(b) Using the *shortcut rule* from [72] to reduce transition t_2 from 3.10a.



(c) Using the *iteration rule* from [72] to reduce the transition t_2' from 3.10b

Figure 3.10: Demonstration of reduction rules from [72].

an approach that allows the possibility of using synthesis rules in an interactive way in the context of building a process modeling engine. The state-based region theory tool in [16] provides a way of synthesizing Petri nets. However, this approach does not allow user interaction, and it does not guarantee soundness of the discovered Petri net. The work in [36] provides a way of interactively synthesizing live and bounded Petri nets using a knitting technique. However, the completeness and soundness of these rules in the context of workflow nets is not guaranteed. Our use of the synthesis rule kit from [58, 73] is two-fold. First, as we have shown, these rules can be easily translated to be used in our context of free-choice workflow nets. Second, these rules guarantee that any sound free-choice workflow net can be synthesized from an initial atomic net.

3.6 Conclusion

In order to enable interactive process modeling, we first identified a need for building blocks which can be used to generate process models. The class of process models chosen was required to support simple representations such as sequences, concurrency and exclusive choices in a process, as well as complex structural representations such as inclusive choices and arbitrary loops. Another important factor taken into consideration was that the process models generated must always be sound. In order to ensure this, we proposed an approach of using the synthesis rules kit from [58], which serves as the building blocks for well-formed free-choice nets. Next, we mapped these synthesis rules from well-formed free-choice nets to sound free-choice workflow nets.

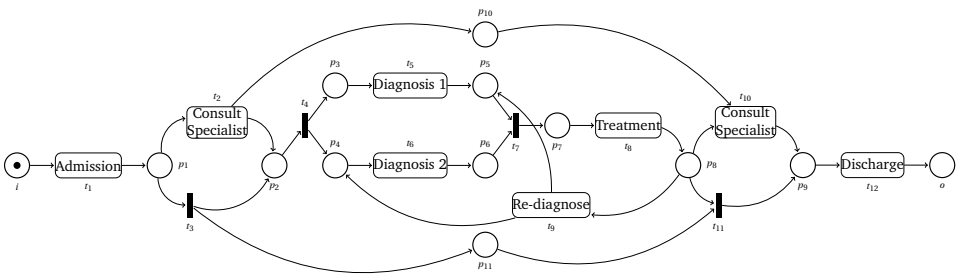


Figure 3.11: Long-term dependency introduced in the care-pathway process by adding places p_{10} and p_{11}. The resulting net is not a free-choice net.

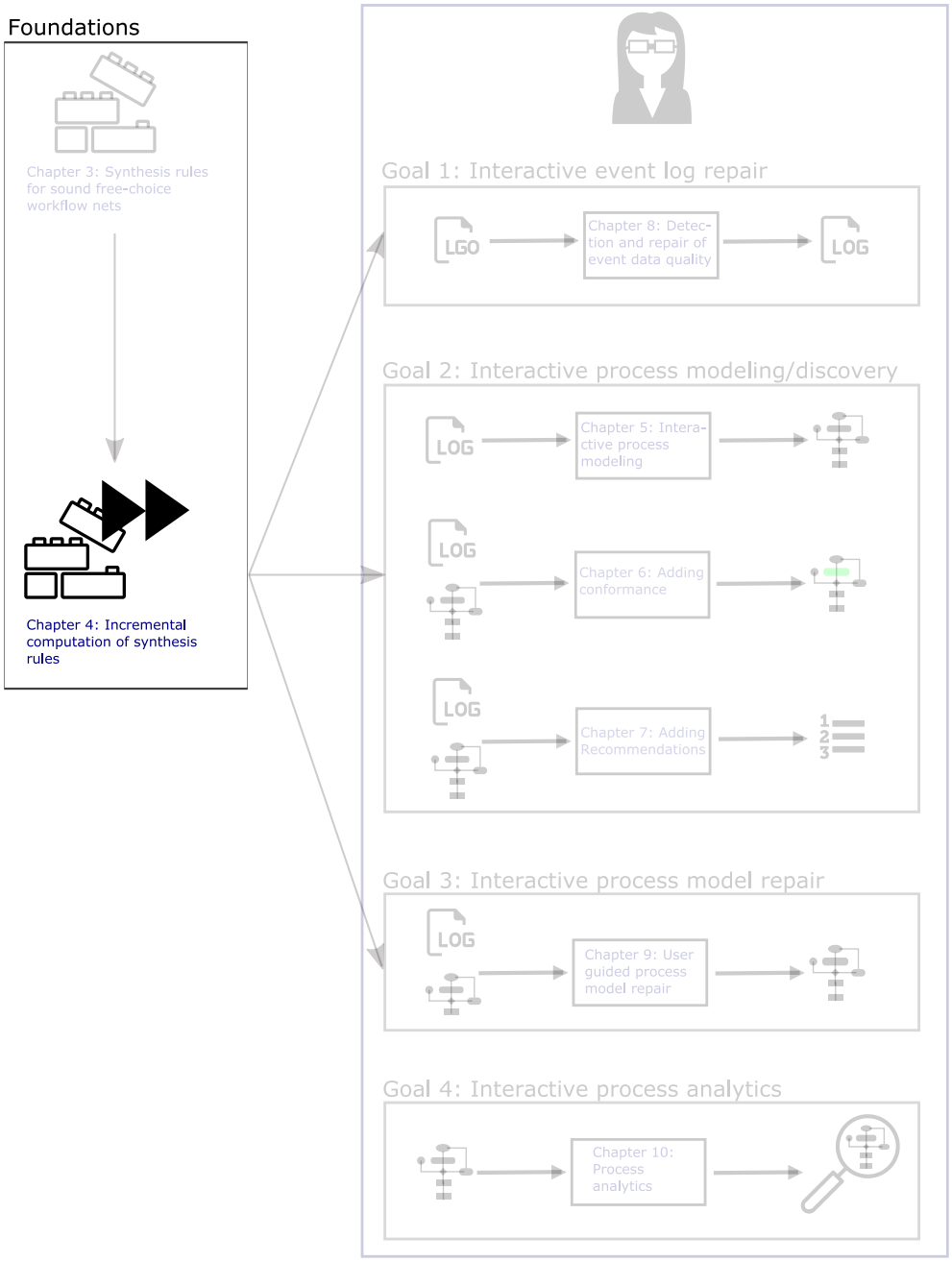
Using the guarantees from [58], we were able to show that the synthesis rules in the context of free-choice workflow nets are correct and complete in order to synthesize any sound free-choice workflow net.

A limitation of using using a free-choice workflow net as the modeling notation is the difficulty in expressing long-term dependencies in a process. For example, consider the variation of the care-pathway process of Figure 1.3, shown in Figure 3.11. The process model from Figure 3.11 places the condition that the second 'Consult Specialist' can be executed if and only if the first 'Consult Specialist' activity was executed. Clearly, the net from Figure 3.11 is not a free-choice net. In order to express long-term dependencies using free-choice net, we would have to duplicate activities, thereby increasing the size of the process model and making it more complicated. However, a similar problem is faced by block-structured notations such as process trees. Therefore, we argue that the advantages of using a free-choice workflow net in combination with synthesis rules such as simple and complex structural representations and guarantee on soundness, outweigh the limitation of difficulty in expressing long-term dependencies.

Using the synthesis rules as the building blocks, we have implemented a robust engine to enable interactive process modeling. This implementation uses a brute force approach to find out all the possible applications of different synthesis rules, with respect to a free-choice workflow net. However, based on the evaluations performed, this approach would be inefficient in an interactive setting, as it took more than 10 milliseconds after 10 synthesis operations. Hence, there is a clear need to compute this synthesis space in a more efficient way, to improve the performance of the system.

Chapter 4

Incremental Computation of Synthesis Rules



This chapter is based on the publication [69].

The approach presented in the previous chapter aimed at providing an "engine" to create a synthesis space for sound free-choice workflow nets that can be used to enable interactive process model construction. Eventually, such a system would be used by an end user for modeling/discovering process models in an interactive fashion. However, as already discussed in the evaluation of Chapter 3, the time taken to compute the synthesis space in a brute-force manner could be very long, and hence undesirable in an interactive setting. In this chapter, we address this concern of computing the synthesis space in a fast-enough way. In order to determine what is *fast-enough*, we consider the following aspects: (i) the desired size (number of nodes) that should be supported in the process model, and (ii) the maximum acceptable response time.

In order to analyze the first aspect, we refer to the "7 Process Modeling Guidelines discussed" discussed in [124]. In particular, we look at guideline G1 and G7, which both deal with the size of a process model. G1 recommends usage of as few elements as possible, whereas G7 recommends decomposing a process model if it contains more than 50 elements. Typically, when the size of a process model increases, the number of errors increase too [125, 138, 175] and the comprehensibility of the process model decreases [123, 126]. Hence, it is in the interest of the modeler to keep the size of the process model of around 50 elements.

In order to analyze the second aspect, we focus on the fields of human-computer-interaction and usability engineering, wherein the effect of response times in the usability of a system have been well researched. [127] provides one of the first works in analyzing the performance issue of an interactive system. Multiple studies [47, 87, 88, 131, 147] have shown that lengthy response times have a detrimental effect on a user's performance and can lead to decreased user productivity. [127, 147] recommend that the ideal response time should be around 2 seconds in an interactive setting. [131] supports this, and further states that a response time of around 1 second keeps a user's flow of thought uninterrupted. Furthermore, [131] suggests that longer response times of about 10 seconds are the limit for keeping a user's attention focused on the task at hand. Finally, response times over 12 seconds are considered disruptive [127, 131, 147] and not recommended. Therefore, it is desirable that the interactive system ideally has response times of around 2 seconds, and does not have response times of more than 12 seconds.

Using the two aspects discussed above, we present an approach to fine-tune the engine discussed in Chapter 3, in order to calculate the synthesis space in an efficient way. In particular, we provide an incremental way of computing the synthesis space. Moreover, we show that the incremental method for computing the synthesis space can be used to correctly compute the complete synthesis space corresponding to a free-choice workflow net.

The outline of the rest of this chapter is as follows. In Section 4.1 we discuss the main problem addressed. In Section 4.2 we present an overview of the approach. In Section 4.3, Section 4.4 and Section 4.5 we provide the details of the proposed approach. In Section 4.6, we compare the incremental approach with the brute-force approach. We conclude this chapter in Section 4.7.

4.1 Problem Definition: Limitations of Brute-force Approach in an Interactive Setting

The synthesis engine recommended in Chapter 3, computes the synthesis space in a brute-force manner. The brute-force approach can take a very long time to compute the synthesis space, even for small-sized nets. Therefore, there is a clear need for fine-tuning the proposed engine, in order to compute the synthesis space in a fast-enough way that is suitable in an interactive setting. Furthermore, the new engine should ensure that the properties of the original engine are still valid. In particular, the new engine should guarantee that precisely all sound free-choice workflow nets can be generated. Using this, we devise the following goal for this chapter:

- How can the engine for generating the building blocks (i.e., the synthesis space) be tuned, such that:
  - process models of reasonable size can be constructed within acceptable response times. Typically, process models containing up to 50 elements are considered optimal [124]. The response time in an interactive setting should be below 10 seconds, and ideally about 2 seconds [127]. Therefore, to be on the higher end, it would be desired that the new engine takes about 2 seconds for computing synthesis space for free-choice workflow nets containing upto 100 nodes.
  - any sound free-choice workflow net can be generated. That is, the synthesis space computed by the new engine should be complete to deduce any sound free-choice workflow net.
  - only sound free-choice workflow net can be generated. That is, the synthesis space computed by the new engine should result only in sound free-choice workflow nets.

In order to address the above goal, we propose an approach to incrementally compute the synthesis space, which is presented in Section 4.2.

4.2 Fast Incremental Computation of the Synthesis Space

The research goal posed in Section 4.1 has three conditions. First, the calculation of the synthesis space in a fast-enough way. Second, a guarantee that any sound free-choice workflow net can be generated. Third, a guarantee that only sound free-choice workflow nets can be generated.

The synthesis space, in a way, directly address the latter two conditions. That is, by the definition of the synthesis space, it contains precisely all sound free-choice workflow nets.

If we take a step back, then we can observe that, to a large extent, $SS(W')$ equals $SS(W)$. We will exploit this fact in our approach by saving the computation time on

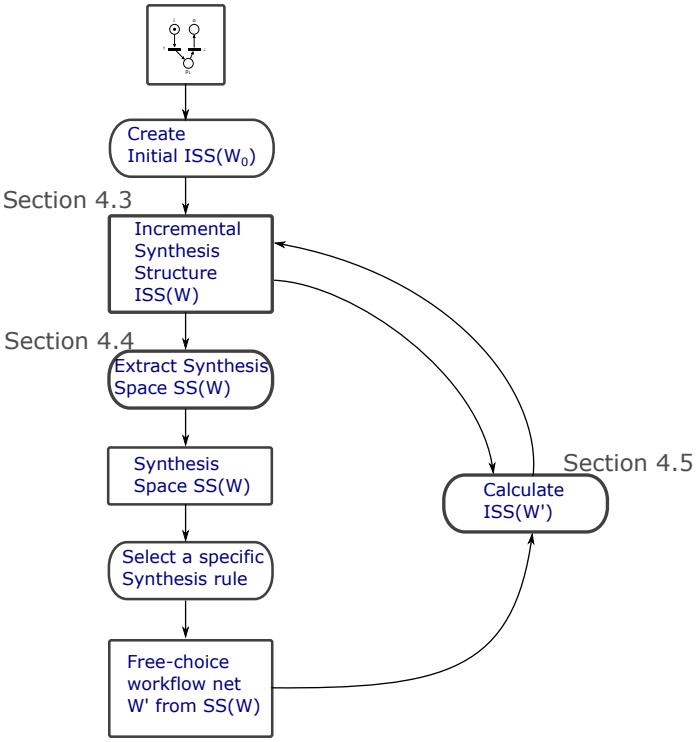


Figure 4.1: Rectangles indicate objects, whereas rounded-rectangles indicate actions. The *Incremental Synthesis Structure* (*ISS*) corresponding to the initial net is calculated in a brute-force way. ISS contains all the possible linear dependencies that could be added to a net resulting in sound free-choice workflow nets, among others. The linear dependencies that result in sound free-choice workflow nets are extracted to populate the synthesis space. Upon selecting a net from the synthesis space, the ISS is updated corresponding to the selected net.

the part where both coincide, and only explore the differences between them. The synthesis space, is generated from the synthesis rules.

In order to address the first condition posed in the research question, we investigate the synthesis rules. Informally speaking, the synthesis rules can be categorized into two categories, local rules and non-local rules.

A rule is considered local if, in order to check the conditions of its application, only the neighborhood of the intended point of application needs to be examined. ψ_A^{WF} is an example of a local rule, as in order to add a new (non-existing) transition t and a new (non-existing) place p to a sound free-choice workflow net $W = (P, T, F, i, o, \top, \bot)$, it suffices to check for a non-empty set of transitions R and a non-empty set of places S which satisfy the condition $R \times S \subseteq F$. It is quite straight-forward to note that the applicability of such local rules can be checked *online* (fast enough way) in an interactive setting, i.e., by checking the neighbors for the presence of arcs between

the set of transitions R and the set of places S.

On the contrary, a rule is considered non-local if it requires an analysis of the complete free-choice workflow net in order to check its applicability. ψ_P^{WF} and ψ_T^{WF} are examples of non-local rules. For example, in order to check if a new place can be added to a sound free-choice workflow net, we need to find if the place is linearly dependent on the entire free-choice workflow net. In other words, it is not (always) possible to check if a rule is applicable by just looking at its neighbors.

Intuitively, it is evident that the synthesis space of local rules can be computed in an efficient way. However, the computation of the synthesis space for non-local rules can be time-consuming. In essence, for every candidate place (transition), we have to check its linear dependence on the entire free-choice workflow net.

In a brute-force approach, we do the following:

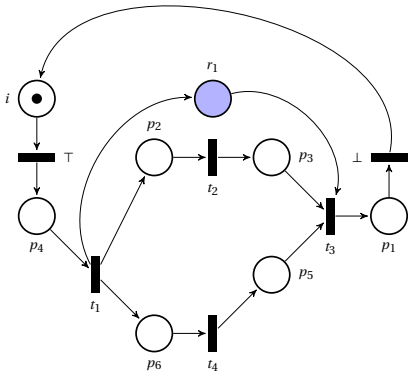
- Generate all possible place (transition) candidates.
- Check for the linear dependence of each place (transition) on the free-choice workflow net.

Clearly, it is computationally expensive to compute the synthesis space from scratch, especially for non-local rules (i.e., linearly dependent rules). In order to overcome this limitation of the brute-force, we introduce an intermediary structure, called the *Incremental Synthesis Structure*, which is used to extract the synthesis space for the non-local rules. After choosing any net from the synthesis space, the incremental synthesis structure is updated (instead of computing from scratch) to contain the information required to extract the synthesis space corresponding to the new net.

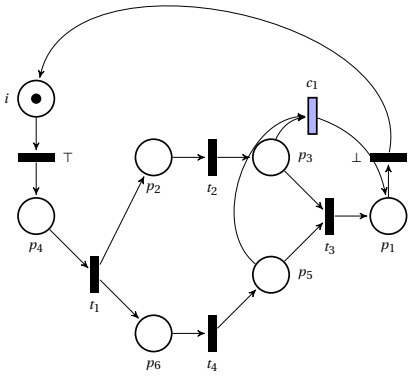
Figure 4.1 provides an overview of our approach. Incremental Synthesis Structure, Extract Synthesis Space and Calculate $ISS(W)$ from Figure 4.1 are the main focus areas of this chapter, which essentially replace the phase of recomputing the synthesis space in a brute-force approach. We begin by discussing and formally defining the incremental synthesis structure in Section 4.3. This is followed by the discussion and proof that given an incremental synthesis structure corresponding to a sound free-choice workflow net, we can extract the synthesis space from it in Section 4.4. We then show how the incremental synthesis structure can be updated after choosing a net from the synthesis space, in order to derive the new incremental synthesis structure corresponding to the newly synthesized net in Section 4.5. It should be noted that by showing that the incremental synthesis structure is correct and complete corresponding to the newly synthesized net, using Section 4.4 it can also be shown that the synthesis space can be extracted from the incremental synthesis structure corresponding to the newly synthesized net.

4.3 Incremental Synthesis Structure

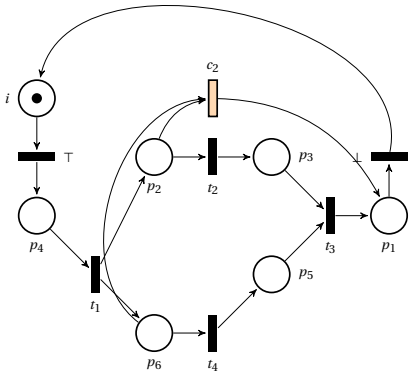
Before discussing the incremental synthesis structure formally, we first give an intuition of what it contains, and how it functions. In essence, the incremental synthesis structure is used to extract the synthesis space corresponding to a given net, as well



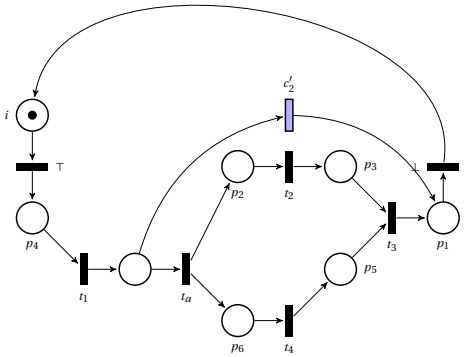
(a) r_1 can be added using ψ_P^{WF}.



(b) c_1 can be added using ψ_T^{WF}.



(c) c_2 results in non-free-choice construct, but is a part of the incremental synthesis structure.



(d) Using ψ_A^{WF} to add t_a and p_a to the net from 4.2c. c_2' which is derived from c_2 results in a free-choice construct.

Figure 4.2: Examples demonstrating candidates of incremental synthesis structure.

as to extract the synthesis spaces of possible future nets derived from the current net. An incremental synthesis structure contains sets **TT** and **PP**, wherein **TT** contains all the possible applications of ψ_T^{WF} rule and **PP** contains all the possible applications of ψ_P^{WF} rule corresponding to a sound free-choice workflow net. Furthermore, both **TT** and **PP** contain some additional information which may not be useful imminently. **TT** and **PP** are divided into two parts as follows:

$$\mathbf{TT} = \boxed{tt_a}\ \boxed{tt_b} \quad \mathbf{PP} = \boxed{pp_a}\ \boxed{pp_b}$$

The first part pp_a (tt_a) of the incremental synthesis structure contains the information for extracting possible candidates that result in free-choice constructs only. Loosely speaking, pp_a (tt_a) contains the information to extract all the candidate nodes which can be added to a sound free-choice workflow net using ψ_P^{WF} (ψ_T^{WF}). For example, Figure 4.2a and Figure 4.2b show nodes r_1 and c_1, which belong to pp_a and tt_a resp. It is clear that both r_1 and c_1 result in free-choice constructs.

The second part of incremental synthesis structure, i.e., pp_b and tt_b, contain the information about the candidates which may result in non-free-choice net constructs. Loosely speaking, pp_b and tt_b contain the information for extracting the nodes which are not applicable directly, but are required for extracting the applications of ψ_P^{WF} and ψ_T^{WF} rules in the future, after the possible usage of the ψ_A^{WF} rule. The intuition behind this mechanism is shown in Figure 4.3. The transition t^n and the place p^n cannot be used directly as they result in non-free-choice constructs. However, after the application of ψ_A^{WF} rule to add p_a and t_a, we can use t^n and p^n to *derive* t^{n+1} and p^{n+1}, which result in free-choice constructs. Consider the transition c_2 from Figure 4.2c. Clearly, adding c_2 would result in a non-free-choice net. However, it should be noted that it is possible to use ψ_A^{WF} rule between a set of transitions ($\{t_1\}$) and all the inputs of c_2 ($\{p_2, p_6\}$). The resulting net after using such a ψ_A^{WF} rule is shown in Figure 4.2d. After using such a ψ_A^{WF} rule, we can then use c_2 to extract a new possible candidate c_2' corresponding to the newly synthesized net. Clearly, c_2' results in a free-choice construct, and hence can be added using ψ_T^{WF} rule.

Figure 4.4b shows an example candidate c_3 which is not a part of either tt_a or tt_b. This is because c_3 results in non-free-choice net construct. Furthermore, it is also not possible to use the ψ_A^{WF} between any set of transitions and the inputs of c_3, i.e., $\{p_2, p_5\}$.

We now define the incremental synthesis structure formally. By re-visiting the de-

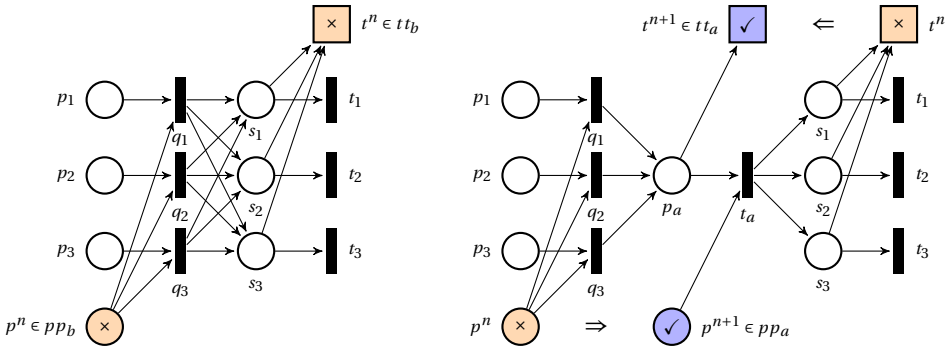
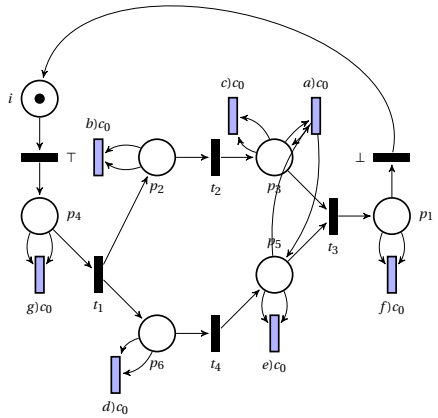
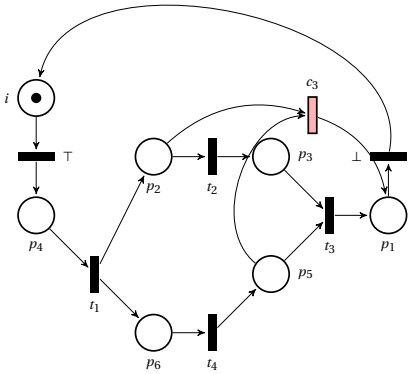


Figure 4.3: How the abstraction rule can be used to extract 'free-choice' candidates from 'non-free-choice' candidates. t_a and p_a are the newly added nodes using ψ_A^{WF}. The candidates t_n and p_n that resulted in non-free-choice net constructs, can be used to extract free-choice net candidates t_{n+1} and p_{n+1} after the usage of ψ_A^{WF}. In other words, we *need* t_n (p_n) to incrementally derive t_{n+1} (p_{n+1}), i.e., without the former, we do not have the latter.



(a) A single vector could result in multiple free-choice candidate nodes. Here, c_0 which is a null vector, results in multiple candidates that could result free-choice nodes (it should be noted that adding one of these candidates to the net, could result in changes in or removal of some of the other candidates).



(b) c_3 results in non-free-choice construct, and is not a part of the incremental synthesis structure.

Figure 4.4: Example showing how a single vector can result in multiple candidates in the incremental synthesis structure, and an invalid vector that cannot be a part of the incremental synthesis structure.

finition of linearly dependent nodes (Definition 26), we know that a node is linearly dependent on a free-choice workflow net, if its corresponding vector representation is linearly dependent on the incidence matrix of the short-circuited free-choice workflow net. In the incremental synthesis structure, instead of dealing directly with candidate places or transitions, we deal with candidate place vectors and candidate transition vectors. These vectors can then be used to generate the candidate place and transition nodes respectively. In essence, the candidate vector indicates the input and output nodes of the candidate node in a free-choice workflow net. We now define the approach to extract a set of pairs from a vector, where each pair contain a set of input nodes and a set of output nodes, of a possible candidate node.

Definition 31 (Input/output node sets from a vector)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ be a free-choice workflow net, and let $\mathbf{c} :^{SC} P \to \{-1, 0, 1\}$ be a $|^{SC}P|$-dimensional vector (Definition 27). Let $P^k = \{s \in P \mid \mathbf{c}(s) = k\}$. Then the set of input/output places for $\mathbf{c}$, denoted as $\mathfrak{N}(\mathbf{c})$, is defined as follows:*

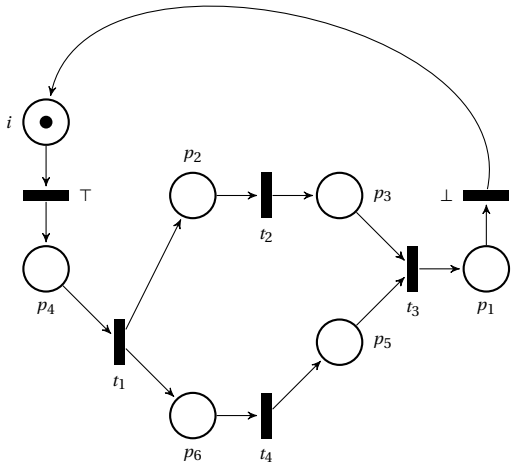
$$\mathfrak{N}(\mathbf{c}) = \{(P_I, P_O) \mid P_I = P^{-1} \cup X \wedge P_O = P^1 \cup X \wedge X \subseteq P^0\}.$$

Basically, if we would have a linearly dependent candidate transition t such that $\bullet t = P_I$ and $t\bullet = P_O$, then $\mathbf{t} = \mathbf{c}$. Thus, $\mathfrak{N}(\mathbf{c})$ gives us all possible matching combinations of presets and postsets, as $P^{-1} = \bullet t \setminus t\bullet, P^1 = t\bullet \setminus \bullet t, P^0 = \bullet t \cap t\bullet$. Similarly, let $\mathbf{r} : T \to \{-1, 0, 1\}$ be a $|T|$-dimensional vector. Let $T^k = \{q \in T \mid \mathbf{r}(q) = k\}$. Then the sets of input/output transitions for $\mathbf{r}$, denoted as $\mathfrak{N}(\mathbf{r})$, is defined as follows:

$$\mathfrak{N}(\mathbf{r}) = \{(T_I, T_O) \mid T_I = T^1 \cup X \wedge T_O = T^{-1} \cup X \wedge X \subseteq T^0\}.$$

In Definition 31 a single vector can result in multiple pairs of input and output node sets, depending on the number of zero's.

For example, consider a vector $\mathbf{c_2}$ from Figure 4.5. Let us ignore the background



	⊤	t_1	t_2	t_3	t_4	⊥	$\mathbf{c_0}$	$\mathbf{c_1}$	$\mathbf{c_2}$	$\mathbf{c_3}$	⋯
i	-1	0	0	0	0	1	0	0	0	0	⋯
p_1	0	0	0	1	0	-1	0	1	1	1	⋯
p_2	0	1	-1	0	0	0	0	0	-1	-1	⋯
p_3	0	0	1	-1	0	0	0	-1	0	0	⋯
p_4	1	-1	0	0	0	0	0	0	0	0	⋯
p_5	0	0	0	-1	1	0	0	-1	0	-1	⋯
p_6	0	1	0	0	-1	0	0	0	-1	0	⋯

$\mathbf{r_1}$	0	1	0	-1	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 4.5: Example short-circuited free-choice workflow net and corresponding incidence matrix. $\mathbf{c_0}$, $\mathbf{c_1}$, $\mathbf{c_2}$ and $\mathbf{r_1}$ are valid ISS vectors, whereas $\mathbf{c_3}$ is not a valid ISS vector (see Figure 4.2)

color of each pair for the time being. From Definition 31, we have

$$\mathfrak{N}(\mathbf{c_2}) = \left\{ \begin{array}{c} (\{p_2, p_6\}, \{p_1\}), \\ (\{p_2, p_6, i\}, \{p_1, i\}), \\ (\{p_2, p_6, p_4\}, \{p_1, p_4\}), \\ (\{p_2, p_6, p_3\}, \{p_1, p_3\}), \\ (\{p_2, p_6, p_5\}, \{p_1, p_5\}), \\ (\{p_2, p_6, i, p_4\}, \{p_1, i, p_4\}), \\ (\{p_2, p_6, i, p_3\}, \{p_1, i, p_3\}), \\ \dots \end{array} \right\}.$$

Basically, for tt_a we are interested in those $(P_I, P_O) \in \mathfrak{N}(\mathbf{c_2})$ for which some transition t exists, such that $\bullet t = P_I$ (to ensure that the net remains free-choice). For tt_b, something similar holds (see Definition 32). Similarly, for the vector r_1 from Figure 4.5, we have

$$\mathfrak{N}(\mathbf{r_1}) = \left\{ \begin{array}{c} (\{t_1\}, \{t_3\}), \\ (\{t_1, \top\}, \{t_3, \top\}), \\ (\{t_1, t_2\}, \{t_3, t_2\}), \\ (\{t_1, t_4\}, \{t_3, t_4\}), \\ (\{t_1, t_5\}, \{t_3, t_5\}), \\ (\{t_1, \bot\}, \{t_3, \bot\}), \\ \dots \end{array} \right\}.$$

For a single candidate vector, there might be multiple input and output nodes possible, which result in free-choice constructs. For example, Figure 4.4a shows the different possible candidate nodes extracted from the vector $\mathbf{c_0}$ of Figure 4.5. Having linked the vectors to possible input and output node sets, we now define the so-called *valid ISS vector*.

Definition 32 (Valid ISS vectors)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ be a sound free-choice workflow net. A $|^{SC}P|$-dimensional vector $\mathbf{c} :^{SC} P \to \mathbb{Z}$ is called a valid ISS column vector in W iff*

1. $\mathbf{c} :^{SC} P \to \{-1, 0, 1\}$ *and* $\mathbf{c}$ *is linearly dependent on the columns of SC(*$\mathbf{W}$*), where SC(*$\mathbf{W}$*) is the incidence matrix of $SC(W)$, and*
2. $\exists (P_I, P_O) \in \mathfrak{N}(\mathbf{c}) \wedge i \notin P_I \cup P_O$*, such that either,*
   (a) $\exists_{q \in T} \bullet q = P_I \wedge P_I, P_O \neq \emptyset$, *cf.* tt_a *or*
   (b) $\exists_{R \subseteq T} R \times P_I \subseteq F \wedge R \neq \emptyset$ *cf.* tt_b

Similarly, a $|T|$-dimensional vector $\mathbf{r} : T \to \mathbb{Z}$ is called a valid ISS row vector in W, iff

1. $\mathbf{r} : T \to \{-1, 0, 1\}$ *and* $\mathbf{r}$ *is linearly dependent on the rows of SC(*$\mathbf{W}$*),*

2. $\exists (T_I, T_O) \in \mathfrak{N}(\mathbf{r})$ *such that either,*

   (a) $\exists_{s \in P} s\bullet = T_O \wedge T_I, T_O \neq \emptyset$ *cf.* pp_a *or*

   (b) $\exists_{S \subseteq P} T_O \times S \subseteq F \wedge S \neq \emptyset$ *cf.* pp_b

The condition (2)(a) of valid ISS vectors from the definition above ensures that the candidate results in free-choice net constructs cf. tt_a and pp_a, whereas condition (2)(b) of valid ISS vectors above ensures that it is possible to use ψ_A^{WF} rule cf. tt_b and pp_b, and thereby the candidate can come into picture at a later point. Furthermore, since we consider the short-circuited version of the net, place o is excluded from the condition (2) of valid ISS column vector above.

Let us look at a couple of examples to demonstrate Definition 32. Consider the vector $\mathbf{r_1}$ from Figure 4.5. Let us compare this vector with the conditions from Definition 32:

1. $\mathbf{r_1}$ is linearly dependent on the incidence matrix as shown in Figure 4.5, as $\mathbf{r_1} = \mathbf{p_2} + \mathbf{p_3}$. Hence $\mathbf{r_1}$ satisfies the condition (1) of Definition 32.

2. Consider the first pair of sets in $\mathfrak{N}(\mathbf{r_1})$, $T_I = \{t_1\}$, $T_O = \{t_3\}$. Furthermore,

   (a) $p_3 \in P, p_3\bullet = T_O = \{t_3\}$. Hence condition (2)(a) of Definition 32 is satisfied.

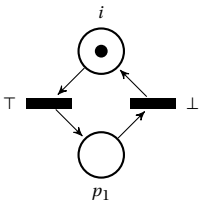
As $\mathbf{r_1}$ satisfies condition (1) and (2)(a) of Definition 32, $\mathbf{r_1}$ is a valid ISS row vector.

Similarly, consider the vector $\mathbf{c_2}$ from Figure 4.5. Let us again compare this vector with the conditions from Definition 32:

1. $\mathbf{c_2}$ is linearly dependent on the incidence matrix as shown in Figure 4.5, as $\mathbf{c_2} = \mathbf{t_2} + \mathbf{t_3} + \mathbf{t_4}$. Hence $\mathbf{c_2}$ satisfies the condition (1) of Definition 32.

2. Consider the first pair of sets in $\mathfrak{N}(\mathbf{c_2})$, $P_I = \{p_2, p_6\}$, $P_O = \{p_1\}$.

   (a) There does not exist any transition which has both p_2 and p_6 as the input. Hence, condition (2)(a) of Definition 32 is violated by $\mathbf{c_2}$.



(a) Short-circuited version of initial free-choice workflow net.

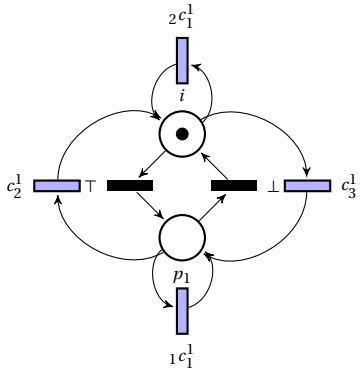
	⊤	⊥
i	-1	1
p_1	1	-1

$\mathbf{r_1^I}$	0	0
$\mathbf{r_2^I}$	1	-1
$\mathbf{r_3^I}$	-1	1

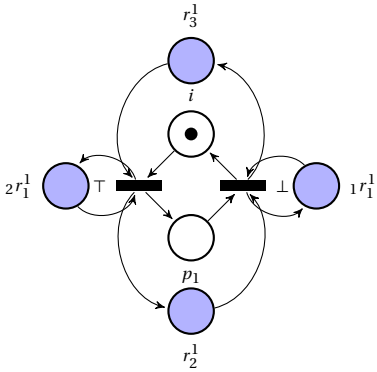
$\mathbf{c_1^I}$	$\mathbf{c_2^I}$	$\mathbf{c_3^I}$
0	-1	1
0	1	-1

(b) Incidence matrix and the incremental synthesis structure corresponding to initial free-choice workflow net.

Figure 4.6: Initial free-choice workflow net and its corresponding incremental synthesis structure.



(a) Transitions derived from possible valid ISS column vectors. It should be noted that $\mathbf{c_1^1}$ results in two nodes $_1c_1^1$ and $_2c_1^1$.



(b) Places derived from possible valid ISS row vectors. It should be noted that $\mathbf{r_1^1}$ results in two nodes $_1r_1^1$ and $_2r_1^1$.

Figure 4.7: Candidate places and transitions corresponding to initial valid ISS vectors.

(b) $R = \{t_1\} \subseteq T \wedge R \times P_I \subseteq F \wedge R \neq \emptyset$. Hence, condition (2)(b) of Definition 32 is satisfied by $\mathbf{c_2}$.

As $\mathbf{c_2}$ satisfies condition (1) and condition (2)(b) of Definition 32, $\mathbf{c_2}$ is a valid ISS column vector.

Having defined *valid ISS vectors*, we now define the incremental synthesis structure.

Definition 33 (Incremental Synthesis Structure (ISS))**.** *Let $W = (P, T, F, i, o, \top, \bot)$ be a sound free-choice workflow net. The incremental synthesis structure is a tuple $ISS(W) =$ (**TT**, **PP**), where:*

1. **TT** *is a set of vectors, such that* $\mathbf{c} \in$ **TT** *iff* **c** *is a valid ISS column vector in W.*

2. **PP** *is a set of vectors, such that* $\mathbf{r} \in$ **PP** *iff* **r** *is a valid ISS row vector in W.*

It should be noted that all the valid ISS vectors are a part of the incremental synthesis structure, according to Definition 33.

We now discuss the valid ISS vectors corresponding to the initial free-choice workflow net. The valid ISS vectors corresponding to the initial free-choice workflow net are shown in Figure 4.6b. It can be trivially verified that valid ISS vectors of the initial free-choice workflow net are correct and complete according to Definition 33. Figure 4.7 shows the possible applications of candidate places and transitions derived using Definition 31 based on the vectors from Figure 4.6b.

Having discussed the incremental synthesis structure, Section 4.4 discusses how it can be used to extract the synthesis space.

4.4 Extracting the Synthesis Space from the Incremental Synthesis Structure

In this section, we discuss how the incremental synthesis structure can be used to extract the synthesis space of a sound free-choice workflow net.

Lemma 1. *Let $W = (P, T, F, i, o, \top, \bot)$ be a sound free-choice workflow net, let $ISS(W) = (\mathbf{TT}, \mathbf{PP})$ and let $W' = (P', T', F', i, o, \top, \bot)$ be such that $(W, W') \in \psi_T^{WF}$ and $T' \setminus T = \{t\}$. Then $\mathbf{t} \in \mathbf{TT}$.*

Similarly, let $W = (P, T, F, i, o, \top, \bot)$ be a sound free-choice workflow net, let $ISS(W) = (\mathbf{TT}, \mathbf{PP})$ and let $W' = (P'', T'', F'', i, o, \top, \bot)$ be such that $(W, W'') \in \psi_P^{WF}$ and $P'' \setminus P = \{p\}$. Then $\mathbf{p} \in \mathbf{PP}$.

Proof. We need to show that $\mathbf{t}$ is a valid ISS column vector in W. By construction of ψ_T^{WF}, we know that:

1. $\mathbf{t} \to \{-1, 0, 1\}$ and $\mathbf{t}$ is linearly dependent on the columns of SC($\mathbf{W}$). Therefore $\mathbf{t}$ satisfies the condition (1) of valid ISS column vector in W.

2. Let $P_I = {}^{W'}\bullet\, t \wedge P_O = t\,{}^{W'}\bullet$. As $SC(W')$ is a short-circuited version of a sound free-choice workflow net, it is strongly connected. Hence, $P_I, P_O \neq \emptyset$. Moreover, since $P_I \subseteq P, P'$ and W and W' are free-choice nets, $\exists q \in T\ {}^{W}\bullet\, q = P_I$. Furthermore, using Definition 31, we know that $(P_I, P_O) \in \mathfrak{N}(\mathbf{t})$. Therefore, $\mathbf{t}$ satisfies the condition (2)(a) of valid ISS column vector in W.

Since $\mathbf{t}$ satisfies conditions (1) and (2)(a) of valid ISS column vector in W, it must be already present in **TT**. Thus, using the vector $\mathbf{t}$ from **TT**, we can extract a part (P_I, P_O), which can be used to add the transition t in the net W to obtain W'.

The proof for the ψ_P^{WF} rule is symmetrical. However, it should be noted that in the case of ψ_P^{WF} rule, there is an additional check required to verify the absence of siphons in a free-choice workflow net. In essence, **PP** also contains candidates that could result in unsound free-choice workflow nets due to the introduction of siphons; these are the candidates which result in well-formed free-choice nets, but not sound free-choice workflow nets. Hence, during the extraction of the synthesis space we need to consider only those candidates which do not result in siphons (without i) in the sound free-choice workflow net. □

From Lemma 1, it can thus be shown that the synthesis space spanned by the ψ_P^{WF} and ψ_T^{WF} rules can be extracted using the incremental synthesis structure. It should be noted that a single valid ISS vector can result in multiple valid candidate nodes in the net, and hence can result in multiple applications of a rule. This is evident from Definition 31. For example, consider the valid ISS column vector $\mathbf{c_1^1}$ from Figure 4.6b. There can be two transitions corresponding to $\mathbf{c_1^1}$ as shown in Figure 4.7a. However, it should be noted that the candidates which have the place i as its input or output, would not be used, as the resulting construct would not be a workflow net. We primarily keep such vectors (along with the vectors which result in siphons in the case of a valid ISS row vector), to keep **TT** and **PP** symmetrical.

Note that the aim is to derive $SS_T(W)$ and $SS_P(W)$ from $ISS(W)$ in an efficient way, such that response times are minimized during editing. In most practical circumstances, we can investigate the net in order to extract only those candidates from **TT** (**PP**), which could result in free-choice constructs. That is, while populating $\mathfrak{N}(\mathbf{c})$ corresponding to a vector $\mathbf{c}$, we consider only those nodes that would result in free-choice constructs, as non-free-choice constructs would not be a part of $SS_T(W)$ ($SS_P(W)$). In other words, we would only explore the tt_a (pp_a) candidates from **TT** (**PP**), thereby speeding-up the process of extracting valid candidates.

We argue that the incremental synthesis structure can be incrementally updated after usage of a synthesis rule, which can then be used to obtain the synthesis space corresponding to the synthesized free-choice workflow net. The incremental synthesis structure corresponding to the initial net W_0 from Figure 4.6a is shown in Figure 4.6b. In the following sections we discuss how the incremental synthesis structure is updated after the usage of each synthesis rule.

4.5 Updates to the Incremental Synthesis Structure

In the following sections, we discuss the changes to the incremental synthesis structure after selecting a net (application of a rule) from the synthesis space. Furthermore, we prove that the changes made are necessary and sufficient to derive the new incremental synthesis structure corresponding to the new net.

In Section 4.4 it was already shown that we can efficiently compute the synthesis space by using the incremental synthesis structure corresponding to a sound free-choice workflow net. Hence, in this section, we only show that the incremental synthesis structure updated after selecting a net from the synthesis space is correct and complete according to Definition 33, and can be done efficiently for nets above 100 nodes or so.

We know that choosing a free-choice workflow net from the synthesis space links to the application of a synthesis rule to a free-choice workflow net. The usage of a synthesis rule on a free-choice workflow net leads to the addition of a new transition and/or new place to the free-choice workflow net. Since the incidence matrix is extended after the application of a synthesis rule, we need to extend the corresponding vectors from the incremental synthesis structure with one dimension corresponding to the newly added node. In order to do so, we first define extending a vector as follows.

Definition 34 (Vector extension $\mathbf{v}|e := k$)**.** *Let* $\mathbf{v} : A \to \mathbb{Q}$ *be an n-dimensional vector, such that* $e \notin A \wedge |A| = n$*. Then* $\mathbf{v}|e := k : A \cup \{e\} \to \mathbb{Q}$ *is an* $(n+1)$*-dimensional vector such that:*

$$\mathbf{v}|e := k(a) = \begin{cases} \mathbf{v}(a), & \text{if } a \in A; \\ k, & \text{otherwise (i.e., } a = e) \end{cases}$$

We now discuss the updates to the incremental synthesis structure, after the application of each type of synthesis rule. We argue that starting with the initial incre-

mental synthesis structure, we can incrementally update the incremental synthesis structure corresponding to any synthesized net.

4.5.1 Updates after the Linearly Dependent Place and Transition Rules

In this section, we discuss the updates in the incremental synthesis structure after the usage of linearly dependent rules. We begin with the ψ_P^{WF} rule. We first define an intermediary structure denoted by $ISS_P(W')$, followed by the proof that $ISS_P(W') = ISS(W')$.

Definition 35 ($ISS_P(W)$)**.** *Let $W = (P, T, F, i, o, \top, \perp)$ and $W' = (P', T', F', i, o, \top, \perp)$ be two sound free-choice workflow nets, where $(W, W') \in \psi_P^{WF}$ and $\{p\} = P' \setminus P$. Let λ be such that $\lambda \cdot SC(\mathbf{W}) = \mathbf{p}$. As $(W, W') \in \psi_P^{WF}$, such a λ exists. Let $ISS(W) = (\mathbf{TT}, \mathbf{PP})$ be the incremental synthesis structure of W. Then we can extract a structure $ISS_P(W) = (\mathbf{PP}_P, \mathbf{TT}_P)$ as follows*:

- $\mathbf{PP}_P = \mathbf{PP}$
- $\mathbf{TT}_P = \bigcup_{\mathbf{c} \in \mathbf{TT}} f_t(\mathbf{c})$ *where,*
  $f_t(\mathbf{c}) = \begin{cases} \{\mathbf{c} | p := \lambda \cdot \mathbf{c}\}, & \text{if } \mathbf{c} | p := \lambda \cdot \mathbf{c} \text{ is a valid ISS column vector in } W' \\ \emptyset, & \text{otherwise} \end{cases}$

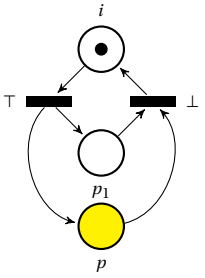
We now show that the intermediary structure defined in Definition 35 is the same as the incremental synthesis structure of the newly synthesized net.

Theorem 10 (Incremental Synthesis Structure after ψ_P^{WF})**.** *Let $W = (P, T, F, i, o, \top, \perp)$ and $W' = (P', T', F', i, o, \top, \perp)$ be two sound free-choice workflow nets, where $(W, W') \in \psi_P^{WF}$ and $\{p\} = P' \setminus P$. Let λ be such that $\lambda \cdot SC(\mathbf{W}) = \mathbf{p}$. As $(W, W') \in \psi_P^{WF}$, such a λ exists. Let $ISS(W) = (\mathbf{TT}, \mathbf{PP})$ be the incremental synthesis structure of W and let $ISS_P(W) = (\mathbf{PP}_P, \mathbf{TT}_P)$ be intermediary structure as defined in Definition 35. Then, the incremental synthesis structure of W' $ISS(W') = (\mathbf{TT}', \mathbf{PP}') = ISS_p(W)$, i.e., $\mathbf{PP}' = \mathbf{PP}_P$ and $\mathbf{TT}' = \mathbf{TT}_P$.*

Proof. We show that Definition 35 is correct and complete separately for $\mathbf{PP}' = \mathbf{PP}_P$ and $\mathbf{TT}' = \mathbf{TT}_P$.

PP′ Since **p** is linearly dependent on the rows of SC(**W**), the ranks of SC(**W**) and SC(**W′**) are the same. Since the rank is unchanged, there cannot be any new linear combinations possible. Since, there are no new transitions added, it can be trivially verified that all the elements of **PP** are valid as-is in the new net W', i.e., $\mathbf{PP} = \mathbf{PP}_P = \mathbf{PP}'$.

TT′ As **p** is linearly dependent on the rows of SC(**W**), the rank is not changed, and no new linear combinations are possible. However, since there is a newly added row, all the vectors from **TT** need to be extended corresponding to the row of the newly added place p. In Definition 35, for a vector **c** this value is chosen to



(a) Short-circuited version of initial free-choice workflow net net, with the newly added place p using ψ_P^{WF}.

	⊤	⊥	$\mathbf{c_1^1}$	$\mathbf{c_2^1}$	$\mathbf{c_3^1}$
i	-1	1	0	-1	1
p_1	1	-1	0	1	-1
p	1	-1	0	1	-1

$\mathbf{r_1^1}$	0	0
$\mathbf{r_2^1}$	1	-1
$\mathbf{r_3^1}$	-1	1

(b) Incidence matrix $SC(\mathbf{W})$ and the incremental synthesis structure corresponding to 4.8a.

Figure 4.8: Theorem 10 in action after using ψ_P^{WF} to add a place p. For example, $\lambda \cdot SC(\mathbf{W}) = \mathbf{p}$, such that $\lambda(i) = 0 \wedge \lambda(p_4) = 1$.

be $\lambda \cdot \mathbf{c}$ (or that vector is discarded). We now show that this is indeed the correct value. Without loss of generality, let us assume that $\mathbf{p}$ is the last row of SC($\mathbf{W}'$):

$$SC(\mathbf{W}') = \begin{pmatrix} SC(\mathbf{W}) \\ \lambda \cdot SC(\mathbf{W}) \end{pmatrix} \; as \; \mathbf{p} = \lambda \cdot SC(\mathbf{W}) \tag{4.1}$$

Let $\mathbf{c} \in \mathbf{TT}$. We know that $\mathbf{c}$ is linearly dependent on the columns of SC($\mathbf{W}$), i.e., for some μ it holds that $\mathbf{c} = SC(\mathbf{W}) \cdot \mu$

We can obtain a corresponding vector which is linearly dependent on the columns of SC($\mathbf{W}'$) as: $SC(\mathbf{W}') \cdot \mu$. If we extend SC($\mathbf{W}'$) with such a vector as the last column, then we get $\begin{pmatrix} SC(\mathbf{W}') & SC(\mathbf{W}') \cdot \mu \end{pmatrix}$. From Equation 4.1, we have

$$\begin{pmatrix} SC(\mathbf{W}) & SC(\mathbf{W}) \cdot \mu \\ \lambda \cdot SC(\mathbf{W}) & \lambda \cdot SC(\mathbf{W}) \cdot \mu \end{pmatrix} = \begin{pmatrix} SC(\mathbf{W}) & \mathbf{c} \\ \lambda \cdot SC(\mathbf{W}) & \lambda \cdot \mathbf{c} \end{pmatrix}, \; as \; we \; know \; that \; SC(\mathbf{W}) \cdot \mu = \mathbf{c}$$

Therefore, for $\mathbf{c}$ to be linearly dependent in SC($\mathbf{W}'$) the value of the row corresponding to the newly added place should be $\lambda \cdot \mathbf{c}$. This is exactly what is done in Definition 35. This is irrespective of the λ chosen. For instance, consider two vectors λ_1 and λ_2 such that $\lambda_1 \cdot SC(\mathbf{W}) = \mathbf{p} = \lambda_2 \cdot SC(\mathbf{W}) \wedge \lambda_1 \neq \lambda_2$. Then $\lambda_1 \cdot \mathbf{tt} = \lambda_2 \cdot \mathbf{c} : \lambda_1 \cdot \mathbf{c} = \lambda_1 \cdot SC(\mathbf{W}) \cdot \mu = \mathbf{p} \cdot \mu = \lambda_2 \cdot SC(\mathbf{W}) \cdot \mu = \lambda_2 \cdot \mathbf{c}$.

□

Figure 4.8 shows the updates to the initial free-choice workflow net and its incidence matrix after adding a new linearly dependent place p using ψ_P^{WF}. This newly added place corresponds to the vector $\mathbf{r_2^1}$ as highlighted in Figure 4.8b. As evident, there are no new valid ISS row vectors possible. Furthermore, the valid ISS column

vectors now contain a value corresponding to the newly added place. Figure 4.8b shows the values corresponding to the newly added place based on Definition 35 and Theorem 10. All the possible nodes based on the incremental synthesis structure of Figure 4.8b are shown in Figure 4.9.

In a similar fashion, the incremental synthesis structure can be updated after the usage of ψ_T^{WF}. We first define an intermediary structure $ISS_T(W)$ as follows.

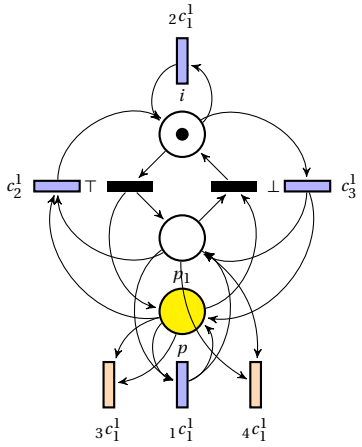
Definition 36 ($ISS_T(W)$)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ and $W' = (P', T', F', i, o, \top, \bot)$ be two sound free-choice workflow nets, where $(W, W') \in \psi_T^{WF}$, where $\{t\} = T' \setminus T$. Let μ be such that $SC(\mathbf{W}) \cdot \mu = \mathbf{t}$. As $(W, W') \in \psi_T^{WF}$, such a μ exists. Let $ISS(W) = (\mathbf{TT}, \mathbf{PP})$ be the incremental synthesis structure corresponding to the net W. Then we can extract an intermediary structure $ISS_T(W) = (\mathbf{PP}_T, \mathbf{TT}_T)$, where:*

- $\mathbf{TT}_T = \mathbf{TT}$
- $\mathbf{PP}_T = \bigcup_{\mathbf{r} \in \mathbf{PP}} f_p(\mathbf{r})$ *where,*

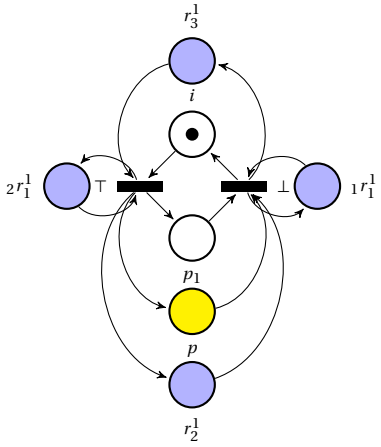
$$f_p(\mathbf{r}) = \begin{cases} \mathbf{r}|t := \mathbf{r} \cdot \mu, & \text{if } \mathbf{r}|t := \mathbf{r} \cdot \mu \text{ is a valid ISS column vector in } W' \\ \emptyset, & \text{otherwise} \end{cases}$$

We now show that the incremental synthesis structure corresponding to the newly synthesized net after the usage of ψ_T^{WF} rule is the same as the intermediary structure of Definition 36.

Theorem 11 (Incremental Synthesis Structure after ψ_T^{WF})**.** *Let $W = (P, T, F, i, o, \top, \bot)$ and $W' = (P', T', F', i, o, \top, \bot)$ be two sound free-choice workflow nets, where $(W, W') \in$*



(a) Transitions derived from possible valid ISS column vectors of Figure 4.8.



(b) Places derived from possible valid ISS row vectors of Figure 4.8.

Figure 4.9: Candidate places and transitions corresponding to Figure 4.8. Note that although we now have more candidate nodes, the number of valid ISS vectors did not increase.

ψ_T^{WF}. Let $ISS(W)$ and $ISS(W')$ be the incremental synthesis structures corresponding to W and W' resp. Let $ISS_T(W)$ be the intermediary structure extracted using Definition 36. Then $ISS_T(W) = ISS(W')$.

Proof. The proof is symmetrical to the proof of Theorem 10. □

Theorem 10 and Theorem 11 show that the incremental synthesis structure (Definition 33) can be correctly and completely computed after the usage of ψ_P^{WF} and ψ_T^{WF} rules. Therefore, using Section 4.4 we can also say that the synthesis space can be correctly and completely extracted after the usage of ψ_P^{WF} and ψ_T^{WF} rules.

It should be noted that, the incremental synthesis structure can not increase in size (i.e., only reduce or stay the same), after the usage of ψ_P^{WF} and ψ_T^{WF} rules. That is, no new candidate vectors are added to the incremental synthesis structure after the usage of ψ_P^{WF} and ψ_T^{WF} rules. In the next sub-section, we discuss the updates to the incremental synthesis structure after the usage of ψ_A^{WF} rule.

4.5.2 Updates after the Abstraction Rule

Following the updates to the incremental synthesis structure after the usage of ψ_P^{WF} and ψ_T^{WF} rules, we now discuss the approach used for extracting the incremental synthesis structure after the usage of ψ_A^{WF} rule.

Before that, we present a few lemmata, which are used to support the theorem for extracting the incremental synthesis structure after the usage of ψ_A^{WF}. We begin by discussing the effect on linear dependencies of valid ISS vectors after the usage of the ψ_A^{WF} rule.

Lemma 2. *Let $W = (P, T, F, i, o, \top, \bot)$ and $W' = (P', T', F', i, o, \top, \bot)$ be two sound free-choice workflow nets, such that $(W, W') \in \psi_A^{WF}$, $\{p\} = P' \setminus P$ and $\{t\} = T' \setminus T$. Let a vector $\mathbf{c}'$ be linearly dependent on the columns of SC($\mathbf{W}'$), such that $\mathbf{c}'(p) = 0$. Consider a vector $\mathbf{c}$, such that $\mathbf{c}' = \mathbf{c}|p := 0$. Then $\mathbf{c}$ is linearly dependent on the columns of SC($\mathbf{W}$) iff $\mathbf{c}'$ is linearly dependent on the columns of SC($\mathbf{W}'$).*

Proof. Without loss of generality, if we assume that the last row of SC($\mathbf{W}'$) corresponds to p and the last column of SC($\mathbf{W}'$) corresponds to t, then from [58] (pg. 139), SC($\mathbf{W}'$) can be decomposed such that: $SC(\mathbf{W}') = \tilde{\mathbf{N}} \cdot A^{-1}$ where

$$\tilde{\mathbf{N}} = \begin{pmatrix} SC(\mathbf{W}) & B \\ 0 \ldots 0 & -1 \end{pmatrix} \textit{ where } \begin{pmatrix} B \\ -1 \end{pmatrix} \textit{ is the last column of SC}(\mathbf{W}')$$

and A^{-1} is a $T \times T$ matrix, such that: $A^{-1}[u, v] = \begin{cases} 1 & \text{if } u = v \\ -1 & \text{if } u = t \wedge v \in {}^{W'}\!\bullet\, p \\ 0 & \text{otherwise} \end{cases}$

=> If $\mathbf{c}'$ is linearly dependent, we have $\mathbf{c}' = SC(\mathbf{W}') \cdot \mu$. Therefore, $\mathbf{c}' = \tilde{\mathbf{N}} \cdot A^{-1} \cdot \mu = \tilde{\mathbf{N}} \cdot \gamma$, where $\gamma = A^{-1} \cdot \mu$. We can re-write this as:

$$\begin{pmatrix} \mathbf{c}'' \\ 0 \end{pmatrix} = \begin{pmatrix} SC(\mathbf{W}) & B \\ 0 \ldots 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Y} \\ \gamma(p) \end{pmatrix} \textit{ where } \gamma = \begin{pmatrix} \mathbf{Y} \\ \gamma(p) \end{pmatrix} \wedge \mathbf{c}' = \begin{pmatrix} \mathbf{c}'' \\ 0 \end{pmatrix}$$

From above, we have $\gamma(p) = 0$. Since $\gamma(p) = 0$ and $\mathbf{c}'(p) = 0$ we have:

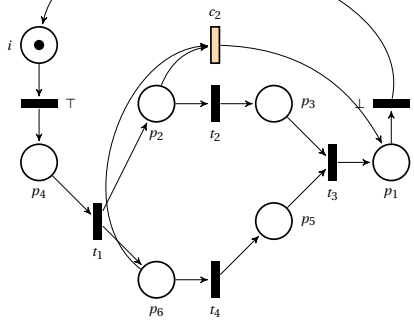
$$\mathbf{c}'' = SC(\mathbf{W}) \cdot \mathbf{Y}$$

Hence, the vector $\mathbf{c}'' = \mathbf{c}$ is linearly dependent on the columns of SC($\mathbf{W}$).

<= Similarly, by following the steps above in reverse, we can show that if $\mathbf{c}$ is linearly dependent on the columns of SC($\mathbf{W}$), then $\mathbf{c}|p := 0$ is linearly dependent on the columns of SC($\mathbf{W}'$).

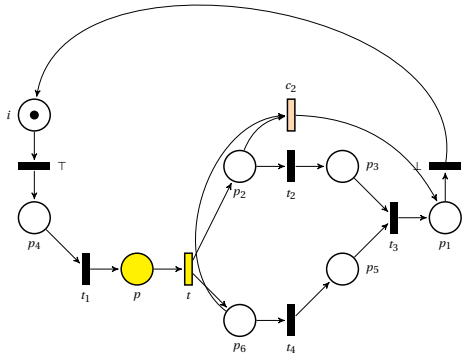
□

A similar argument can be made about the valid ISS row vectors, using Lemma 2 symmetrically. Figure 4.10 shows a demonstration of Lemma 2. In Figure 4.10a, we have $\mathbf{c_2} = \mathbf{t_2} + \mathbf{t_3} + \mathbf{t_4}$. After applying the ψ_A^{WF} rule in Figure 4.10b, it is easy to see that $\mathbf{c_2}$ remains a valid ISS column vector, as $\mathbf{c_2} = \mathbf{t_2} + \mathbf{t_3} + \mathbf{t_4}$.



	⊤	t_1	t_2	t_3	t_4	⊥	$\mathbf{c_2}$
i	-1	0	0	0	0	1	0
p_4	1	-1	0	0	0	0	0
p_2	0	1	-1	0	0	0	-1
p_3	0	0	1	-1	0	0	0
p_1	0	0	0	1	0	-1	1
p_5	0	0	0	-1	1	0	0
p_6	0	1	0	0	-1	0	-1

(a) Example of a sound free-choice workflow net and corresponding incidence matrix. c_2 is a valid ISS column vector.



	⊤	t_1	t_2	t_3	t_4	⊥	t	$\mathbf{c_2}$
i	-1	0	0	0	0	1	0	0
p_4	1	-1	0	0	0	0	0	0
p_2	0	0	-1	0	0	0	1	-1
p_3	0	0	1	-1	0	0	0	0
p_1	0	0	0	1	0	-1	0	1
p_5	0	0	0	-1	1	0	0	0
p_6	0	0	0	0	-1	0	1	-1
p	0	1	0	0	0	0	-1	0

(b) After using ψ_A^{WF}. The extended version of c_2 remains a valid ISS column vector, such that the value corresponding to newly added place p is 0.

Figure 4.10: Example of Lemma 2, after application of ψ_A^{WF} rule.

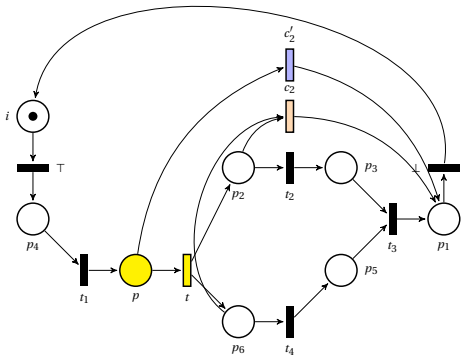
We now show how free-choice constructs from valid ISS vectors (cf. tt_a and pp_a) can be extracted using the non-free-choice constructs of valid ISS vectors (cf. tt_b and pp_b) after usage of the ψ_A^{WF} rule.

Lemma 3. *Let W be a sound free-choice workflow net, let $\mathbf{c}$ be a valid ISS column vector in W satisfying condition (1) of Definition 32. Let $(P_I, P_O) \in \mathfrak{N}(\mathbf{c})$, and let $R \subseteq T$ be such that $R \times P_I \subseteq F \wedge R \neq \emptyset$. Let $(W, W') \in \psi_A^{WF}$, such that $\{t\} = T' \setminus T \wedge \{p\} = P' \setminus P \wedge t^{\bullet^{W'}} = P_I$. Then $\mathbf{c}|p := 0 + \mathbf{t}$ is a valid ISS column vector in W', and the pair $(\{p\}, P_O) \in \mathfrak{N}(\mathbf{c}|\mathbf{p} := \mathbf{0} + \mathbf{t})$ satisfies the condition (2)(a) of Definition 32.*

Proof. By construction of ψ_A^{WF}, we know the values of the column vector corresponding to t in SC($\mathbf{W}'$) are: $\mathbf{t}(s) = \begin{cases} -1, & \text{if } s = p \\ 1, & \text{if } s \in P_I \\ 0, & \text{otherwise} \end{cases}$

As $\mathbf{c}$ is linearly dependent on the columns of $SC(\mathbf{W})$, from Lemma 2, we know that a vector $\mathbf{c}|p := 0$ is linearly dependent on the columns of $SC(\mathbf{W}')$, and hence a vector $\mathbf{c}' = \mathbf{c}|p := 0 + \mathbf{t}$ is also linearly dependent on columns of $SC(\mathbf{W}')$. The values of such a vector $\mathbf{c}'$ are as follows:

$$\mathbf{c}'(s) = \begin{cases} 0, & \text{when } s \in P_I \setminus P_O \quad \text{as } \mathbf{c}|p := 0 s = -1 \wedge \mathbf{t}(s) = 1 \\ 1, & \text{when } s \in P_O \setminus P_I \quad \text{as } \mathbf{c}|p := 0 s = 1 \wedge \mathbf{t}(s) = 0 \\ 1, & \text{when } s \in P_I \cap P_O \quad \text{as } \mathbf{c}|p := 0 s = 0 \wedge \mathbf{t}(s) = 1 \\ -1, & \text{when } s = p \quad \text{as } \mathbf{c}|p := 0 s = 0 \wedge \mathbf{t}(s) = -1 \\ 0, & \text{otherwise} \quad \text{as } \mathbf{c}|p := 0 s = 0 \wedge \mathbf{t}(s) = 0 \end{cases}$$



	$\top$	t_1	t_2	t_3	t_4	$\bot$	t	$\mathbf{c_2}$	$\mathbf{c'_2}$
i	-1	0	0	0	0	1	0	0	0
p_4	1	-1	0	0	0	0	0	0	0
p_2	0	0	-1	0	0	0	1	-1	0
p_3	0	0	1	-1	0	0	0	0	0
p_1	0	0	0	1	0	-1	0	1	1
p_5	0	0	0	-1	1	0	0	0	0
p_6	0	0	0	0	-1	0	1	-1	0
p	0	1	0	0	0	0	-1	0	-1

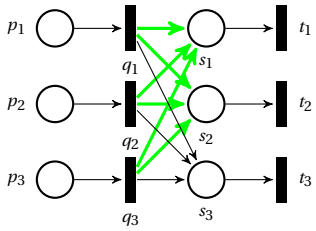
Figure 4.11: Example illustrating Lemma 3. $\mathbf{c_2}$ is a valid ISS row vector in the net before using ψ_A^{WF} (i.e., Figure 4.10a), and $(\{p_2, p_6\}, \{p_1\}) \in \mathfrak{N}(\mathbf{c_2})$. c'_2 is a valid ISS row vector, where $\mathbf{c'_2} = \mathbf{c_2} + \mathbf{t}$ and $(\{p\}, \{p_1\}) \in \mathfrak{N}(\mathbf{c'_2})$.

As $\mathbf{c}' :^{SC} P' \to \{-1, 0, 1\}$ and $\mathbf{c}'$ is linearly dependent on the columns of $SC(\mathbf{W}')$, it satisfies condition (1) of Definition 32. We have a pair $(\tilde{P_I}, \tilde{P_O}) \in \mathfrak{N}(\mathbf{c}')$, such that $\tilde{P_I} = \{p\}$ and $\tilde{P_O} = P_O$. Since $^{W'}\bullet t = \tilde{P_I}$, $\mathbf{c}'$ is valid according to condition (2)(a) of Definition 32 for the pair $(\{p\}, P_O)$. □

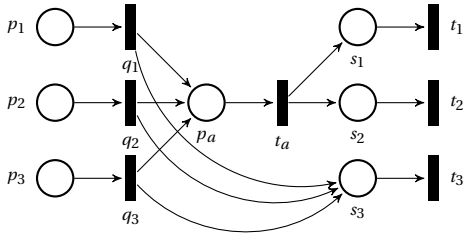
A similar argument can be made about a row vector which is valid according to condition (2)(b) of Definition 32. Figure 4.11 shows an example of Lemma 3.

Lemma 4. *Let $W = (P, T, F, i, o, \top, \bot)$ be a sound free-choice workflow net. Let $(W, W') \in \psi_A^{WF}$ such that $\{p\} = P' \setminus P \wedge \{t\} = T' \setminus T$, and let $^{W'}\bullet p = R \wedge t^{W'}\bullet = S$. Let $\mathbf{c}'$ be a valid ISS column vector in the net W'. Let $(P_I, P_O) \in \mathfrak{N}(\mathbf{c}')$ which satisfies either condition (2)(a) or (2)(b) of Definition 32. Then, $p \in P_I \implies S \cap P_I = \emptyset$, and $p \in P_O \implies S \cap P_O = \emptyset$.*

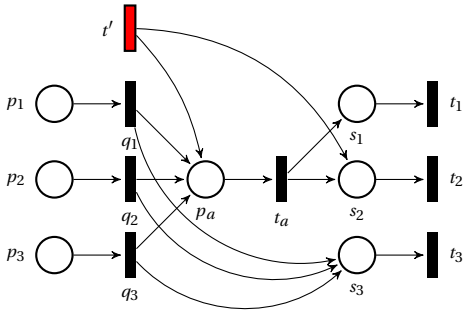
Proof. The crux of this proof lies in the fact that well-formedness implies S-coverability in a short-circuited well-formed free-choice workflow net. As a result, all synthesis rules preserve S-coverability. For both condition (2)(a) and condition (2)(b) (with Lemma 3) from Definition 32, it can be shown that any S-component covering a place



(a) Fragment of a sound free-choice workflow net. ψ_A^{WF} can be used between $\{q_1, q_2, q_3\}$ and $\{s_1, s_2\}$.



(b) Fragment from 4.12a after using ψ_A^{WF} to add p_a and t_a.



(c) Fragment of 4.12b after using ψ_T^{WF} to add t' with some input, and output $\{p_a, s_2\}$. No S-component covers s_2.

Figure 4.12: Example demonstrating Lemma 4.

$s \in S$ also has to cover the place p. Adding a transition which has both s and p as inputs (outputs), would necessarily break all S-components involving the place s, leaving s uncovered. As a result, if p is an input (output) of a transition to be added, then s can not be an input (output) of this transition as well (see Figure 4.12). □

Using T-coverability, we can prove a similar theorem for R and t. Lemmata 2-4 which are related to the usage of ψ_A^{WF} and the discussions on the impact of the incremental synthesis structure allows us to formulate the theorem for updating the prior elements and creating new elements in the incremental synthesis structure after the usage of ψ_A^{WF}.

We begin by first defining an intermediary structure $ISS_A(W)$.

Definition 37 ($ISS_A(W)$)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ and $W' = (P', T', F', i, o, \top, \bot)$ be two sound free-choice workflow nets, such that $(W, W') \in \psi_A^{WF}$. Let $\{p\} = P' \setminus P \wedge \{t\} = T' \setminus T$. Let $\overset{W'}{\bullet} p = R \wedge t \overset{W'}{\bullet} = S$. Let $ISS(W) = (\mathbf{TT}, \mathbf{PP})$ be the incremental synthesis structure corresponding to W. Then we can extract an intermediary structure $ISS_A(W) = (\mathbf{PP}_A, \mathbf{TT}_A)$, where*

- $\mathbf{TT}_A = \bigcup_{\mathbf{c} \in \mathbf{TT}} f_a(\mathbf{c})$ *where,*

$$f_a(\mathbf{c}) = \begin{array}{ll} \{\mathbf{c}|p := 0 & | \ \mathbf{c}|p := 0 \text{ is a valid ISS column vector in } W' \} \cup \\ \{\mathbf{c}|p := 0 + \mathbf{t} & | \ \mathbf{c}|p := 0 + \mathbf{t} \text{ is a valid ISS column vector in } W' \} \cup \\ \{\mathbf{c}|p := 0 - \mathbf{t} & | \ \mathbf{c}|p := 0 - \mathbf{t} \text{ is a valid ISS column vector in } W' \} \end{array}$$

- $\mathbf{PP}_A = \bigcup_{\mathbf{r} \in \mathbf{PP}} f_a(\mathbf{r})$ *where,*

$$f_a(\mathbf{r}) = \begin{array}{ll} \{\mathbf{r}|t := 0 & | \ \mathbf{r}|t := 0 \text{ is a valid ISS row vector in } W' \} \cup \\ \{\mathbf{r}|t := 0 + \mathbf{p} & | \ \mathbf{r}|t := 0 + \mathbf{p} \text{ is a valid ISS row vector in } W' \} \cup \\ \{\mathbf{r}|t := 0 - \mathbf{p} & | \ \mathbf{r}|t := 0 - \mathbf{p} \text{ is a valid ISS row vector in } W' \} \end{array}$$

We now show that the incremental synthesis structure corresponding to the newly synthesized net is the same as the intermediary structure from Definition 37.

Theorem 12 (Incremental Synthesis Structure after ψ_A)**.** *Let $W = (P, T, F, i, o, \top, \bot)$ be a sound free-choice workflow net and let $ISS(W) = (\mathbf{TT}, \mathbf{PP})$ be its incremental synthesis structure. Let $W' = (P', T', F', i, o, \top, \bot)$ be a sound free-choice workflow net, such that $(W, W') \in \psi_A^{WF}$. Let $\{p\} = P' \setminus P \wedge \{t\} = T' \setminus T$. Let $\overset{W'}{\bullet} p = R \wedge t \overset{W'}{\bullet} = S$. Let $ISS_A(W) = (\mathbf{PP}_A, \mathbf{TT}_A)$ be the intermediary structure extracted using Definition 37 and let $ISS(W') = (\mathbf{PP}', \mathbf{TT}')$ be the incremental synthesis structure of W'. Then $ISS(W') = ISS_A(W)$, i.e., $\mathbf{PP}' = \mathbf{PP}_A$ and $\mathbf{TT}' = \mathbf{TT}_A$.*

Proof. We assume that **TT** and **PP** are correct and complete, and by induction, show that $\mathbf{TT}_A$ and $\mathbf{PP}_A$ are then correct and complete corresponding to W'. Definition 37 is correct by construction as only the valid ISS vectors are added to $\mathbf{TT}_A$ and $\mathbf{PP}_A$. Therefore, we only need to show that Definition 37 is complete according to Definition 33 for W'.

Table 4.1: Values corresponding to an arbitrary place s.

$s = p$	$s \in \tilde{P_I}$	$s \in \tilde{P_O}$	$s \in S$	$\mathbf{t}$	$\mathbf{c'_1}$	$\mathbf{c'_2}$	$\mathbf{c'_3}$	$\mathbf{c'_4}$
✓				-1	0	-1	1	0
			✓	1	0	0	0	0
		✓		0	1	1	1	1
	✓			0	-1	-1	-1	-1
		✓	✓	0	✗	1	✗	1
	✓		✓	0	✗	✗	-1	-1
	✓	✓		0	0	0	0	0
	✓	✓	✓	0	✗	✗	✗	0
				0	0	0	0	0

It is trivial to check the presence of the zero vector in the incremental synthesis structure of W'. For non-zero vectors, we take an arbitrary valid ISS column vector $\mathbf{c}' \in \mathbf{TT}'$, and show that we can obtain it from some vector $\mathbf{c} \in \mathbf{TT}$ in Definition 37. Since $\mathbf{c}'$ is a valid ISS column vector in the incremental synthesis structure of W', we know that there exists a pair $(P_I, P_O) \in \mathfrak{N}(\mathbf{c}')$, which satisfies either condition (2)(a) or condition (2)(b) of Definition 32. We prove the completeness based on the presence (or absence) of the newly added place p in P_I and P_O. We can have four cases: *(i)* $p \in P_I \wedge p \in P_O$, in this case, we refer to the vector as $\mathbf{c'_1}$, *(ii)* $p \in P_I \wedge p \notin P_O$, in this case, we refer to the vector as $\mathbf{c'_2}$, *(iii)* $p \notin P_I \wedge p \in P_O$, in this case, we refer to the vector as $\mathbf{c'_3}$, and *(iv)* $p \notin P_I \wedge p \notin P_O$, in this case, we refer to the vector as $\mathbf{c'_4}$.

Let $\tilde{P_I} = P_I \setminus \{p\}$ and $\tilde{P_O} = P_O \setminus \{p\}$. The value corresponding to an arbitrary place s in the vectors $\mathbf{c'_1}$,$\mathbf{c'_2}$,$\mathbf{c'_3}$,$\mathbf{c'_4}$ and $\mathbf{t}$ is shown in Table 4.1. For example, for a place s, such that $s \neq p \wedge s \in \tilde{P_I} \wedge s \in S \wedge s \notin \tilde{P_O}$ the value of $\mathbf{c'_3}(s)$ is -1, according to Table 4.1. The values for some of the elements are not present. These are the impossible cases which would otherwise violate Lemma 4. For example, for $\mathbf{c'_2}$, the value corresponding to $s \neq p \wedge s \in \tilde{P_I} \wedge s \in S \wedge s \notin \tilde{P_O}$ is empty. This is because in the case (ii) corresponding to $\mathbf{c'_2}$, $p \in P_I$. Hence $S \cap \tilde{P_I} = \emptyset$. We now show the proof of completeness on a case basis:

(i) $p \in P_I \wedge p \in P_O$ **($\mathbf{c'_1}$)** Consider a vector $\mathbf{c}$, such that $\mathbf{c'_1} = \mathbf{c}|p := 0$. Using Lemma 2, we can say that $\mathbf{c}$ satisfies the condition (1) of Definition 32 in W. For all places except p, $\mathbf{c}$ has the same values as $\mathbf{c'_1}$. Hence from Table 4.1, we know that $(S \cup \tilde{P_I}, S \cup \tilde{P_O}) \in \mathfrak{N}(\mathbf{c})$ in W. We show that this pair satisfies the condition (2)(b) of Definition 32 in W, and thus $\mathbf{c}$ should be present in $\mathbf{TT}$, using which we can get $\mathbf{c'_1}$. Since $\mathbf{c'_1}$ is a valid ISS column vector, we have two cases:

$\mathbf{c'_1}$ satisfies condition (2)(a) Then there exists a $q \in T'$ such that $q \overset{W'}{\bullet} = P_I$. From the construction of ψ_A^{WF}, we know that $t \overset{W'}{\bullet} = \{p\}$. As $p \in P_I$ and the net is free-choice net, we conclude that $P_I = \{p\}$. Hence, we have $\tilde{P_I} = \emptyset$ (see Figure 4.13). Hence the set of input places is only S. However, by the construction of ψ_A^{WF}, $R \times S \subseteq F$ in W. Hence $(S \cup \tilde{P_I}, S \cup \tilde{P_O})$ satisfies the condition (2)(b) of Definition 32 in the net W.

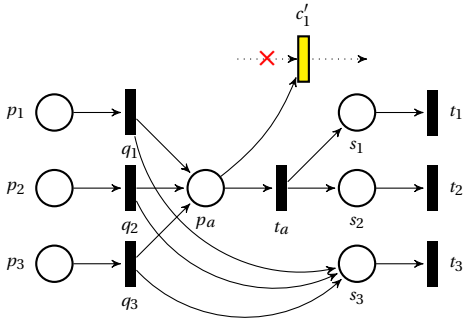
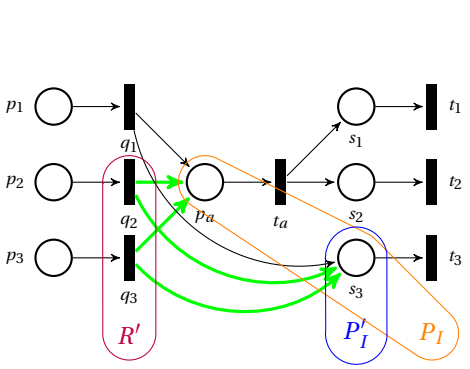


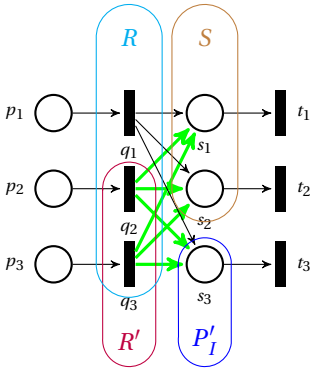
Figure 4.13: Fragment from Figure 4.12b after the application of ψ_A^{WF}. If c_1' is extracted from a valid ISS column vector which satisfies condition (1) and (2)(a) of Definition 32, then if $p_a \in P_I$, implies that $P_I = \{p_a\}$.

$\mathbf{c_1'}$ satisfies condition (2)(b) Then there exists a $R' \subseteq T'$ such that $R' \times P_I \subseteq F'$, i.e., $R' \times (\{p\} \cup \tilde{P}_I) \subseteq F'$. As $R' \times \{p\} \subseteq F'$, we know that $R' \subseteq R$. Hence, by the construction of ψ_A^{WF}, we know that in the net W, $R' \times (\tilde{P}_I \cup S) \subseteq F$ (see Figure 4.14a and Figure 4.14b). Hence $(S \cup \tilde{P}_I, S \cup \tilde{P}_O)$ satisfies the condition (2)(b) of Definition 32 in the net W. This is demonstrated in Figure 4.14

(ii) $p \in P_I \wedge p \notin P_O$ **($\mathbf{c_2'}$)** Consider a vector $\mathbf{c}$, such that $\mathbf{c_2''} = \mathbf{c}|p := 0$. Using Lemma 2, we know that $\mathbf{c}$ satisfies condition (1) of Definition 32 in W. For all the places except p, $\mathbf{c}$ has the same values as $\mathbf{c_2''}$. Hence from Table 4.1, we can get a pair $(P_I' \cup S, P_O) \in \mathfrak{N}(\mathbf{c})$ in W. We show that this pair is valid according to condition (2)(b) in Definition 32, and thus using $\mathbf{c}$ we can get $\mathbf{c_2''}$, from which we can get



(a) Fragment of Figure 4.12b. A pair (P_I, P_O) that satisfies the condition (2)(b) of Definition 32.



(b) Fragment from 4.14a in the original net before using ψ_A^{WF}, such that $R' \subseteq R \wedge S = \{s_1, s_2\} \wedge P_I' = \{s_3\}$.

Figure 4.14: Demonstration for c_1' which satisfies condition (2)(b) of Definition 32.

$\mathbf{c}'_2$. Since $\mathbf{c}'_2$ is a valid ISS column vector, we have two cases:

$\mathbf{c}'_2$ is valid according to (2)(a) Then there exists a $r \in T'$ such that ${}^{W'}\bullet r = P_I$. From the construction of ψ_A^{WF}, we know that ${}^{W'}\bullet r = \{p\}$. As $p \in P_I$ and the net is free-choice, we conclude that $P_I = \{p\}$. Hence, $P'_I = \emptyset$. Since $R \times S \subseteq F$, the pair $(P'_I \cup S, P_O) \in \mathfrak{N}(\mathbf{c})$ satisfies the condition (2)(b) of Definition 32 in W.

$\mathbf{c}'_2$ is valid according to (2)(b) Then there exists a $R' \subseteq T'$ such that $R' \times P_I \subseteq F'$, i.e., $R' \times (\{p\} \cup P'_I) \subseteq F'$. As $R' \times \{p\} \subseteq F'$, we know that $R' \subseteq R$. By the construction of ψ_A^{WF}, we know that $R' \times P'_I \subseteq F$, if $R' \times P'_I \subseteq F'$. Hence the pair $(P'_I \cup S, P_O) \in \mathfrak{N}(\mathbf{c})$ satisfies the condition (2)(b) of Definition 32, in the net W.

(iii) $p \notin P_I \wedge p \in P_O$ **($\mathbf{c}'_3$)** Consider a vector $\mathbf{c}$, such that $\mathbf{c}''_3 = \mathbf{c}|p := 0$. Using Lemma 2, we know that $\mathbf{c}$ satisfies the condition (1) of Definition 32. For all the places except p, $\mathbf{c}$ has the same values as $\mathbf{c}''_3$. Hence, from Table 4.1, we can get a pair $(P'_I, P'_O \cup S) \in \mathfrak{N}(\mathbf{c})$ in W. We have, $P_I = P'_I$ and (P_I, P_O) satisfies either condition (2)(a) or (2)(b) from Definition 32 in W'. By the construction of ψ_A^{WF}, we can say that $(P_I, P'_O \cup S)$ is also valid in W, as the input of both pairs is the same. Hence $\mathbf{c}$ is a valid ISS column vector in W. Using $\mathbf{c}$, we can get $\mathbf{c}''_3$ from which we can get $\mathbf{c}'_3$.

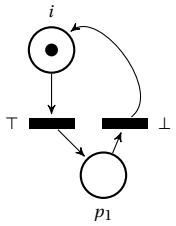
(iv) $p \notin P_I \wedge p \notin P_O$ **($\mathbf{c}'_4$)** Consider a vector $\mathbf{c}$, such that $\mathbf{c}''_4 = \mathbf{c}|p := 0$. Using Lemma 2, we can say that $\mathbf{c}$ valid according to condition (1) of Definition 32. Furthermore, if (P_I, P_O) is valid according to either (2)(a) or (2)(b) from Definition 32 in W', then it is also valid in W by the construction of ψ_A^{WF}. Therefore, $\mathbf{c}$ is a valid ISS column vector in W. Thus using $\mathbf{c}$ we can get $\mathbf{c}'_4$.

□

The correctness and completeness of **PP** is similar and can also be proven incrementally.

4.5.3 Example of Synthesizing Free-choice Workflow Nets using the Incremental Synthesis Structure

In this section, we show the changes to the incremental synthesis structure upon the usage of ψ_A^{WF}, ψ_P^{WF} and ψ_T^{WF} rules: starting with the initial net and the corresponding incremental synthesis structure as shown in Figure 4.15. Consecutively each figure shows the rule used, changes in the net and the incremental synthesis structure in yellow, or red - if the corresponding change results in invalid candidates. Furthermore, each valid ISS vector follows the following naming convention: $\mathbf{m}^{\mathbf{k}}_{\mathbf{j}}$, where k indicates the sequence of net, starting with the initial free-choice workflow net which has a sequence of 1. The value of j is used to link valid ISS vector across different sequences.

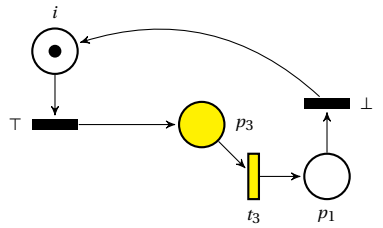


(a) Short-circuited version of initial free-choice workflow net net.

	⊤	⊥	$\mathbf{c_1^1}$
i	-1	1	0
p_1	1	-1	0
$\mathbf{r_1^1}$	0	0	
$\mathbf{r_2^1}$	1	-1	
$\mathbf{r_3^1}$	-1	1	

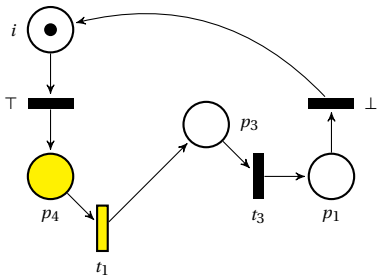
(b) Incidence matrix and the incremental synthesis structure corresponding to initial free-choice workflow net. For the sake of convenience we do not show any valid ISS column vectors which have i in its input (-1) or output (1).

Figure 4.15: Initial free-choice workflow net and its corresponding incremental synthesis structure.



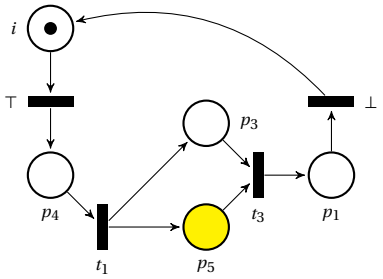
	⊤	⊥	t_3	$\mathbf{c}_1^2 = \mathbf{c}_1^1 \mid p_3 := 0$	$\mathbf{c}_2^2 = \mathbf{c}_1^1 \mid p_3 := 0 \; +\mathbf{t_3}$	$\mathbf{c}_3^2 = \mathbf{c}_1^1 \mid p_3 := 0 \; -\mathbf{t_3}$
i	-1	1	0	0	0	0
p_1	0	-1	1	0	-1	1
p_3	1	0	-1	0	1	-1
$\mathbf{r}_1^2 = \mathbf{r}_1^1 \mid t_3 := 0$	0	0	0			
$\mathbf{r}_2^2 = \mathbf{r}_2^1 \mid t_3 := 0$	1	-1	0			
$\mathbf{r}_3^2 = \mathbf{r}_3^1 \mid t_3 := 0$	-1	1	0			
$\mathbf{r}_4^2 = \mathbf{r}_1^1 \mid t_3 := 0 \; +\mathbf{p_3}$	1	0	-1			
$\mathbf{r}_5^2 = \mathbf{r}_1^1 \mid t_3 := 0 \; -\mathbf{p_3}$	-1	0	1			
$\mathbf{r}_6^2 = \mathbf{r}_2^1 \mid t_3 := 0 \; +\mathbf{p_3}$	2	-1	-1			
$\mathbf{r}_7^2 = \mathbf{r}_2^1 \mid t_3 := 0 \; -\mathbf{p_3}$	0	-1	1			
$\mathbf{r}_8^2 = \mathbf{r}_3^1 \mid t_3 := 0 \; +\mathbf{p_3}$	0	1	-1			
$\mathbf{r}_9^2 = \mathbf{r}_3^1 \mid t_3 := 0 \; -\mathbf{p_3}$	-2	1	1			

Figure 4.16: After adding a place p_3 and a transition t_3 using ψ_A^{WF}. The changes in incremental synthesis structure are shown using Theorem 12. The yellow row and column indicate the newly added place and transition. The red colored cells indicate that the newly derived vector is not a valid ISS vector.



	⊤	⊥	t_3	t_1	$\mathbf{c_1^3}$	$\mathbf{c_2^3}$	$\mathbf{c_3^3}$	$\mathbf{c_4^3}$	$\mathbf{c_5^3}$	$\mathbf{c_6^3}$	$\mathbf{c_7^3}$	$\mathbf{c_8^3}$	$\mathbf{c_9^3}$
i	-1	1	0	0	0	0	0	0	0	0	0	0	0
p_1	0	-1	1	0	0	1	-1	0	0	1	1	-1	-1
p_3	0	0	-1	1	0	-1	1	1	-1	0	-2	2	0
p_4	1	0	0	-1	0	0	0	-1	1	-1	1	-1	1
$\mathbf{r_1^3}$	0	0	0	0									
$\mathbf{r_2^3}$	1	-1	0	0									
$\mathbf{r_3^3}$	-1	1	0	0									
$\mathbf{r_4^3}$	1	0	-1	0									
$\mathbf{r_5^3}$	-1	0	1	0									
$\mathbf{r_7^3}$	0	-1	1	0									
$\mathbf{r_8^3}$	0	1	-1	0									
$\mathbf{r_{10}^3}$	1	0	0	-1									
$\mathbf{r_{11}^3}$	-1	0	0	1									
$\mathbf{r_{12}^3}$	2	0	-1	-1									
$\mathbf{r_{13}^3}$	0	0	-1	1									
$\mathbf{r_{14}^3}$	0	0	1	-1									
$\mathbf{r_{15}^3}$	-2	0	1	1									
$\mathbf{r_{16}^3}$	2	-1	0	-1									
$\mathbf{r_{17}^3}$	0	-1	0	1									
$\mathbf{r_{18}^3}$	1	-1	1	-1									
$\mathbf{r_{19}^3}$	-1	-1	1	1									
$\mathbf{r_{20}^3}$	0	1	0	-1									
$\mathbf{r_{21}^3}$	-2	1	0	1									
$\mathbf{r_{22}^3}$	1	1	-1	-1									
$\mathbf{r_{23}^3}$	-1	1	-1	1									

Figure 4.17: After adding a place p_4 and a transition t_1 using ψ_A^{WF}. Vectors such as $\mathbf{r_{18}^3}$ and $\mathbf{r_{22}^3}$ are not valid ISS row vectors, because neither do they result in free-choice constructs, nor could the transitions with the corresponding value -1 be used to apply the ψ_A^{WF} rule. Hence, both the conditions (2)(a) and (b) of Definition 32 are violated. The newly derived vectors are as follows: $\mathbf{c_1^3} = \mathbf{c_1^2}|p_4 := 0$, $\mathbf{c_2^3} = \mathbf{c_2^2}|p_4 := 0$, $\mathbf{c_3^3} = \mathbf{c_3^2}|p_4 := 0$, $\mathbf{c_4^3} = \mathbf{c_1^2}|p_4 := 0 + \mathbf{t_1}$, $\mathbf{c_5^3} = \mathbf{c_1^2}|p_4 := 0 - \mathbf{t_1}$, $\mathbf{c_6^3} = \mathbf{c_2^2}|p_4 := 0 + \mathbf{t_1}$ and so on.

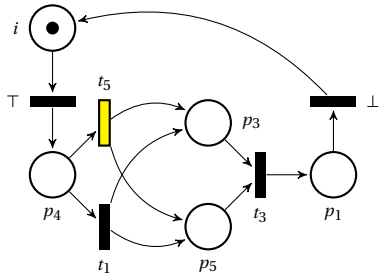


	⊤	⊥	t_3	t_1
i	-1	1	0	0
p_1	0	-1	1	0
p_3	0	0	-1	1
p_4	1	0	0	-1
p_5	0	0	-1	1

$\mathbf{c}_1^4$	$\mathbf{c}_2^4$	$\mathbf{c}_3^4$	$\mathbf{c}_4^4$	$\mathbf{c}_5^4$	$\mathbf{c}_6^4$	$\mathbf{c}_9^4$
0	0	0	0	0	0	0
0	1	-1	0	0	1	-1
0	-1	1	1	-1	0	0
0	0	0	-1	1	-1	1
0	-1	1	1	-1	0	0

$\mathbf{r}_1^4$	0	0	0	0
$\mathbf{r}_2^4$	1	-1	0	0
$\mathbf{r}_3^4$	-1	1	0	0
$\mathbf{r}_4^4$	1	0	-1	0
$\mathbf{r}_5^4$	-1	0	1	0
$\mathbf{r}_7^4$	0	-1	1	0
$\mathbf{r}_8^4$	0	1	-1	0
$\mathbf{r}_{10}^4$	1	0	0	-1
$\mathbf{r}_{11}^4$	-1	0	0	1
$\mathbf{r}_{13}^4$	0	0	-1	1
$\mathbf{r}_{14}^4$	0	0	1	-1
$\mathbf{r}_{17}^4$	0	-1	0	1
$\mathbf{r}_{20}^4$	0	1	0	-1

Figure 4.18: After adding a place p_5 using ψ_P^{WF}. This place is extracted from r_{13}^4 (r_{13}^3 in the previous incremental synthesis structure). Since a new place is introduced, there are no new valid ISS row vectors possible. Hence all the prior valid ISS row vectors from Figure 4.17 are also present here. Similarly, no new valid ISS column vectors are possible either. However, prior valid ISS column vectors need to be updated, corresponding to the newly added place p_5, as highlighted in yellow based on Theorem 10.



	⊤	⊥	t_3	t_1	t_5
i	-1	1	0	0	0
p_1	0	-1	1	0	0
p_3	0	0	-1	1	1
p_4	1	0	0	-1	-1
p_5	0	0	-1	1	1

$\mathbf{c}_1^5$	$\mathbf{c}_2^5$	$\mathbf{c}_3^5$	$\mathbf{c}_4^5$	$\mathbf{c}_5^5$	$\mathbf{c}_6^5$	$\mathbf{c}_9^5$
0	0	0	0	0	0	0
0	1	-1	0	0	1	-1
0	-1	1	1	-1	0	0
0	0	0	-1	1	-1	1
0	-1	1	1	-1	0	0

$\mathbf{r}_1^5$	0	0	0	0	0
$\mathbf{r}_2^5$	1	-1	0	0	0
$\mathbf{r}_3^5$	-1	1	0	0	0
$\mathbf{r}_4^5$	1	0	-1	0	0
$\mathbf{r}_5^5$	-1	0	1	0	0
$\mathbf{r}_7^5$	0	-1	1	0	0
$\mathbf{r}_8^5$	0	1	-1	0	0
$\mathbf{r}_{10}^5$	1	0	0	-1	-1
$\mathbf{r}_{11}^5$	-1	0	0	1	1
$\mathbf{r}_{13}^5$	0	0	-1	1	1
$\mathbf{r}_{14}^5$	0	0	1	-1	-1
$\mathbf{r}_{17}^5$	0	-1	0	1	1
$\mathbf{r}_{20}^5$	0	1	0	-1	-1

Figure 4.19: After adding a transition t_4 using ψ_T^{WF}. This transition is extracted from c_4^5 (c_4^4 in the previous incremental synthesis structure). Since a new linearly dependent transitions is added to the net, no new valid ISS vectors are possible. However, the prior valid ISS row vectors may be needed to be updated, based on the value corresponding to the newly added transition t_5, as highlighted in yellow based on Theorem 11.

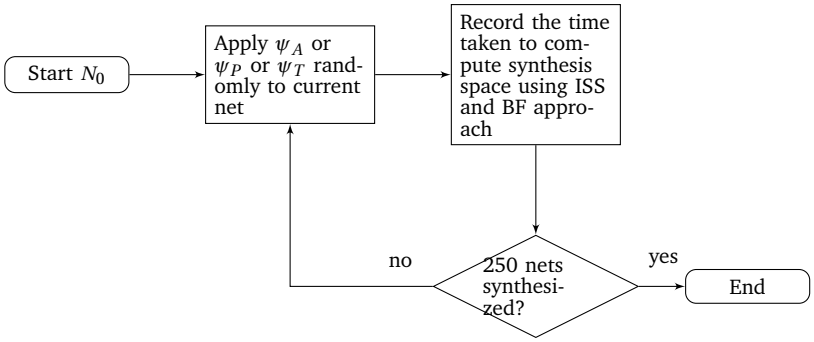


Figure 4.20: The experimental setup to test the performance of incremental synthesis structure (ISS) based approach vs Brute-force (BF) approach (repeated 30 times in total).

4.6 Evaluation

In this section, we evaluate the usage of incremental synthesis structure (ISS) to compute the synthesis space for a given net. Unlike the brute-force (BF) approach, the time taken to compute the synthesis space using the ISS approach is not *always* exponential in the number of nodes in the net. Under certain conditions, the number of linearly dependent rules that could be added to the net could be exponential, and hence the time taken to compute ISS could be exponential too. Figure 4.21 shows an example of such a net. Suppose we would like to add a new place using the ψ_P^{WF} rule. We can do so by selecting any combination of transitions from each *path*, i.e., t_1 or t_2, and t_3 or t_4, and t_5 or t_6 as the input transitions, and $\perp$ as the output transition for the newly added place. This clearly shows the number of applications of the linear dependency rules could be exponential based on the size of the net. That is, if we consider the number of paths to be n, and the number of transitions in each path to be m, then the total number of linearly dependent places that could be added is m^n. However, we argue that in practical circumstances, it suffices to use ISS to compute the synthesis space, as it is generally much faster than the brute-force approach. More importantly, in a brute-force approach we have to check the existence

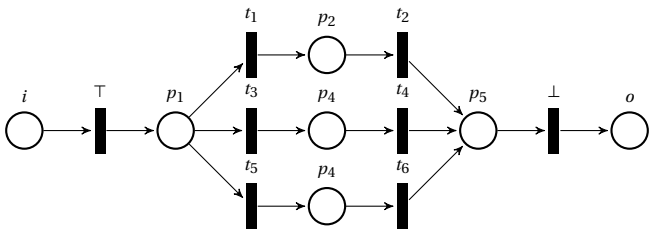


Figure 4.21: An example demonstrating how the number of linearly dependent places that could be added to the net can be exponential to the size of the net.

of linear dependency for each of the vectors, which has an added complexity. Whereas in the ISS approach, the linear dependency is already guaranteed by the inherent mechanism used to compute the vectors incrementally.

We now compare the time taken to compute the synthesis spaces by ISS and BF approaches under the same conditions and machine, by using the same solver. In order to do so, starting with the initial net W_0, a random sound free-choice workflow net was synthesized by using a random synthesis rule. Another random synthesis rule was applied on the synthesized net, to obtain a new random sound free-choice workflow net. This was repeated until 250 random synthesis rules were applied. At any point, each of the three synthesis rules ($\psi_A^{WF}, \psi_P^{WF}, \psi_T^{WF}$) had equal chance of being chosen. This essentially relates to choosing a random sound free-choice workflow net from the synthesis space.

Figure 4.20 shows the experimental set-up. After applying a synthesis rule, the time taken for computing the synthesis space using the brute-force approach and the incremental synthesis structure-based approach were recorded. In an interactive setting, for example, for editing business process models, the response times of the system should be as short as possible. Hence calculating the synthesis space was aborted if the time taken to compute the synthesis space exceeded 20,000 milliseconds. This only concerned the brute-force, as for the incremental synthesis structure-based approach, the computation times were always below 20000 milliseconds. This experiment was repeated 30 times in total. That is, starting with the initial net, random synthesis rules were applied 250 times, for a total of 30 times.

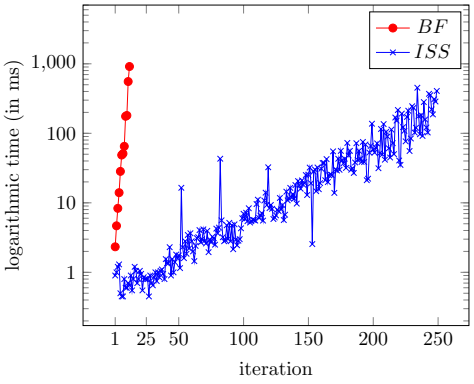
The average time taken to compute the synthesis space using the brute-force approach and the incremental synthesis structure-based approach at each synthesis iteration is plotted in Figure 4.22a. It should be noted that the time scale is logarithmic. In order to compare the brute-force approach and the incremental synthesis structure-based approach effectively, only time averages below 1000 ms are plotted.

For the brute-force approach, the time taken to compute the synthesis space rises quickly and exponentially. Just after 10 synthesis iterations, the average time taken to compute the synthesis space using the brute-force approach is more than 5000 milliseconds. The high computation times for the brute-force approach can be attributed to the following factors:

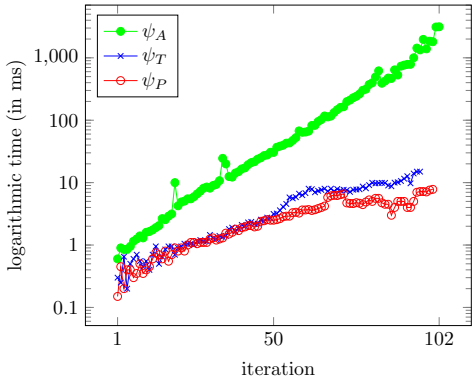
1. As the number of nodes grows with each synthesis iteration, the number of possible permutations grows exponentially. In the brute-force approach, calculating all the possible applications of ψ_T^{WF} and ψ_P^{WF} rules requires a full exploration of all possible vector permutations, in order to generate all the possible candidates.

2. For each of the generated candidates, it is verified if the net would remain a free-choice net. That is, it is verified if there exists at least one pair in Definition 31, using which the net would remain a free-choice net. All the candidates that violate this condition are removed.

3. For each of the remaining candidates, it is verified if the candidate is linearly dependent on the incidence matrix of the net.



(a) Average time taken for brute-force (BF) vs incremental synthesis structure (ISS) based approach after 250 random synthesis iterations (repeated 30 times). Brute-force computation was stopped after 10 synthesis iterations as it took longer than 5000 ms.



(b) Average computation times for incremental synthesis structure after usage of ψ_P^{WF}, ψ_T^{WF} and ψ_A^{WF} rules resp.

Figure 4.22: Performance analysis of the brute-force approach and the incremental synthesis structure approach.

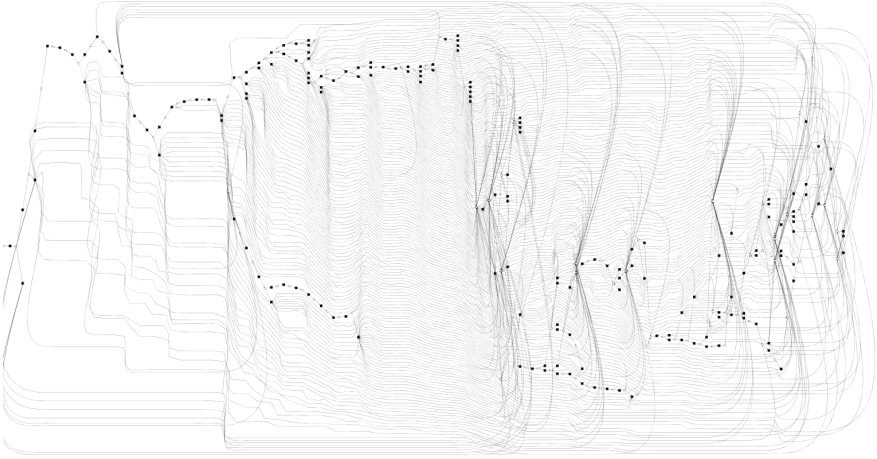


Figure 4.23: One of the sound free-choice workflow nets synthesized after 250 synthesis iterations. This example illustrates that it is not practical to look at such large models. Hence the response times of our approach are acceptable under real-life circumstances.

Clearly, as the number of nodes grows, the number of permutations grows too. Moreover, validating the linear dependence of multiple vectors becomes inefficient rather quickly.

Compared to this, the incremental synthesis structure-based approach proposed in this chapter is still exponential but much faster. This can be attributed to the fact that, unlike the brute-force approach, in the incremental synthesis structure-based approach, we do not have to compute all the possible permutations for any given net. More importantly, the third step for the verification of linear dependence is not required in the incremental synthesis structure-based approach. This is due to the fact that the incremental synthesis structure results only in linearly dependent vectors after the usage of ψ_A^{WF}, ψ_P^{WF} and ψ_T^{WF} rules. Therefore, the incremental synthesis structure-based approach mainly deals only with the non-expensive step 2, when compared to the brute-force approach.

The extraction of all the valid ψ_T^{WF} and ψ_P^{WF} rules from the incremental synthesis structure is rather trivial. Practically, for any non-zero valid ISS column vector in **TT** to result in ψ_T^{WF}; it can be quickly verified if the places that have a value of -1 corresponding to them can result in a free-choice node (along with some possible places which have a value of 0 corresponding to them). A similar argument can be made for the row vectors from **PP**.

When computing the synthesis space for the incremental synthesis structure-based approach, times are highly variable. This is due to the fact that depending on the type of rule used, the computation times of the incremental synthesis structure vary, as shown in Figure 4.22b. For example, if a ψ_A^{WF} rule is used, then additional candidates are generated, each of which needs to be validated according to Definition 32. It should be noted that, the linear dependence condition of these candidate vectors is valid by construction.

Compared to ψ_A^{WF}, after the usage of ψ_P^{WF} and ψ_T^{WF} rules, no new candidates are generated. On the contrary, prior candidates are updated and invalid candidates may be removed. Hence, the computation of the synthesis space after usage of ψ_P^{WF} and ψ_T^{WF} rules is much faster compared to the computation of the synthesis rules after the usage of the ψ_A^{WF} rule. It should be noted that in Figure 4.22b we plot the averages corresponding to each type of synthesis rule. Contrary to this, in Figure 4.22a we plot the averages across all the rules.

As illustrated by Figure 4.22a, the time taken for computing the synthesis space grows exponentially with the brute-force approach. The time taken to compute the synthesis space using the incremental synthesis structure-based approach also grows exponentially, however this growth is more gradual and acceptable for our application scenario. For example, while synthesizing larger nets, the incremental synthesis structure-based approach took, on an average, less than 1 second even after 250 iterations, i.e., after applying 250 synthesis rules starting with the initial net W_0. It can be argued that in practical circumstances, the free-choice workflow nets of sizes over 250 nodes may already be too complicated for the user to comprehend, as shown in Figure 4.23. It is clear that the proposed approach is much faster and outperforms the brute-force approach, and hence is suited for synthesizing very large sound free-choice workflow nets in an interactive way (i.e., by having good response times).

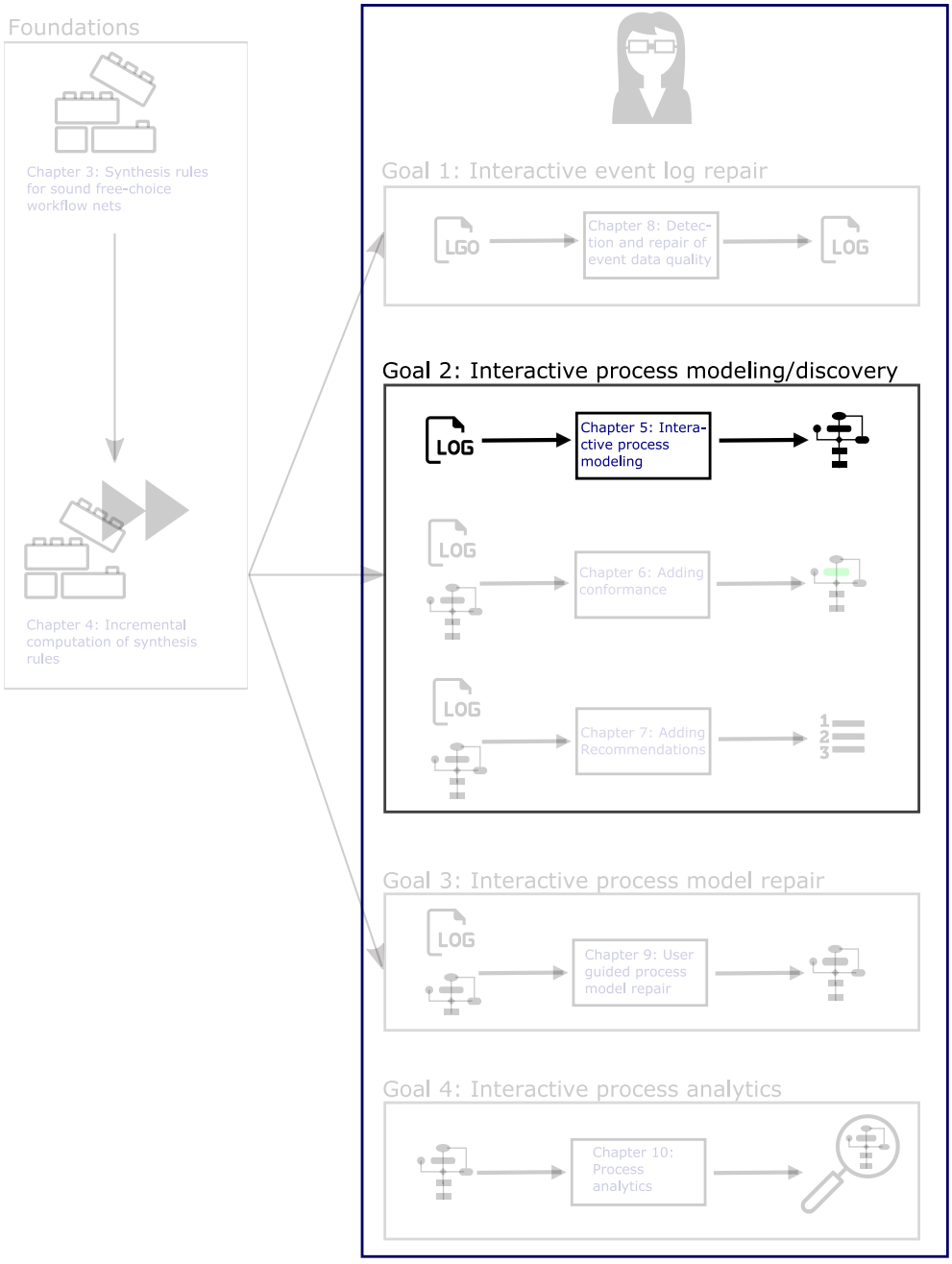
4.7 Conclusion

In this chapter, we introduced a way to compute the synthesis space in an efficient way. The proposed approach is based on a meta-structure called the incremental synthesis structure, which can be used to extract the synthesis space corresponding to any sound free-choice workflow net. The incremental synthesis structure is updated incrementally upon using a synthesis rule. We have shown that the incremental synthesis structure contains sufficient information to extract all the elements of the synthesis space. Furthermore, it was shown that the updates performed to the incremental synthesis structure after the usage of a synthesis rule are correct and complete.

In order to test the scalability of the system, very large sound free-choice workflow nets, more than four times the size of a regular sound free-choice workflow net, were synthesized. It was observed that the incremental synthesis structure-based approach was able to compute the synthesis space within a second for free-choice workflow nets which are twice as big as a regular sound free-choice workflow net containing more than 100 nodes. These response times are clearly acceptable in an interactive setting, as initially discussed in Section 4.1. Overall, the incremental synthesis structure-based approach was clearly able to outperform the brute-force approach for computing the synthesis space.

Chapter 5

Interactive Process Modeling using the Synthesis Engine and the Event Logs



This chapter is based on the publication [67].

Traditionally, process discovery is a phase in the BPM lifecycle wherein an analyst first interviews different actors involved in a process, and then combines the domain knowledge along with the information extracted through interviews, in order to manually model an end-to-end process model. However, in such settings, the process analyst may miss out on the real behavior of the process. The actors involved in the process may *think* they perform tasks in a certain way or in a certain order, but in reality may do something else.

With the advent of digitization, the execution histories of processes are recorded by information systems and can be extracted as event logs. Using these event logs, the phase of process discovery can be automated using process mining techniques, in particular process discovery algorithms, in order to discover process models.

Most process discovery algorithms aim to discover process models *automatically* by learning certain *patterns* from an event log, hence bypassing the analyst. Automated process discovery algorithms work well in a setting where the event log contains all the necessary information (for example, noise-free, complete), required by the discovery algorithm, and the language of the underlying process model is about the same as the language of the process models discovered by the discovery algorithm. However, in many real-world scenarios this is not the case.

Thus, in a practical setting, automated process discovery algorithms typically suffer from the following issues:

1. The process models discovered by the automated discovery algorithms are constrained by the vocabulary of the language used for representing the model, i.e., the representational bias [160]. Most of the automated process discovery techniques cannot discover duplicate activities in a process model (i.e., activities that occur more than once). Some of the process discovery algorithms, such as the α algorithm [169], cannot discover silent activities (i.e., skippable activities). Discovery techniques such as the inductive miner [110] can only discover block-structured process models. As already mentioned in Chapter 1, it is vital that the language used for representing a process model can support representation of sequences, inclusive and exclusive choices, silent activities, duplicate activities, loops and non-block-structured constructs.

2. Many discovery algorithms may discover process models which are unsound. As discussed previously in Chapter 2, a sound process model guarantees an option to execute each activity in the process at least once and the ability to reach the final state (thereby terminating the process) from any valid reachable state. In practical settings, unsound process models are often not interesting to the analyst and hence are discarded. Ideally, the discovery algorithm should limit the search space to only sound process models. However, for many discovery algorithms such as the heuristic miner [185], or the split miner [9], this is not the case, and the discovered process model can be unsound.

3. The process model discovered by an automated process discovery technique may explain the event logs extremely well, but may still be completely incomprehensible to the end user. Therefore, it is imperative to enable the analyst

to have control over the process model being discovered, and thereby also enabling incorporation of domain knowledge during process discovery. In order to discover comprehensible process models, it is desirable to follow an incremental and structured approach of process modeling [43, 124], which is normally not the case in automated process discovery techniques.

In order to address the limitations of automated process discovery techniques and manual process modeling, we propose an approach for interactively constructing a process model, by bridging the gap between manual and automated process discovery. The first and the second limitation of the automated process discovery techniques discussed above are already addressed by the use of free-choice workflow nets and synthesis rules, as shown in Chapter 3 and Chapter 4. The third limitation discussed above deals with a user's comprehensibility of a process model. This also links back to the traditional method of modeling a process model based on domain knowledge. Hence, in this chapter, we focus on enabling interactive process modeling by allowing the user to make informed decisions using the information from the event logs.

Figure 5.1 shows an overview of the approach proposed in this chapter. In this chapter, we particularly address the following two sub-goals of Goal 2 in Chapter 1:

- First, how can the *feedback from the event log* be used in order to assist the

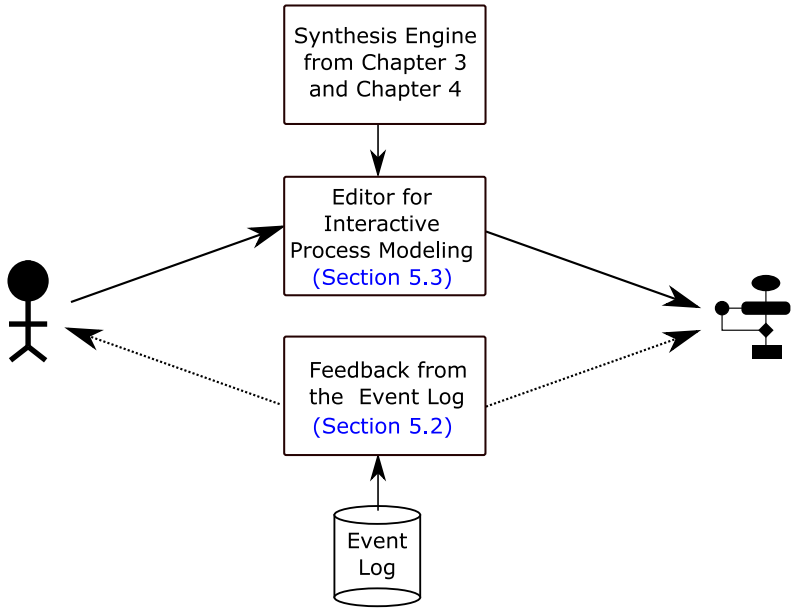


Figure 5.1: Visual depiction of an interactive editing process. The user can edit/discover sound process models based on the editing engine described in Chapter 3 and Chapter 4. guided by the information from the event log. The editing engine is based on the synthesis rules from Chapter 3, and the synthesis rules are computed incrementally using the approach suggested in Chapter 4.

user in decision-making. This subgoal caters to the process mining aspect of interactive process modeling. It can be further divided as follows:

  - Extracting information from the event log: This is similar to the way automated process discovery extracts patterns from the event log in order to make decisions. In our case, we would like to abstract information from the event log, which can be used by a user, instead of an automated process discovery technique, to construct a process model.
  - Presenting the information from the event log to the user: The information extracted from the event log should be presented to a user in a meaningful way, so that the user can effectively combine the domain knowledge with the information from the event log to construct a process model.

- Second, enabling the interactive editing of process models in an incremental and structured manner. In order to address this, we use the synthesis rules based synthesis engine proposed in Chapter 3 and Chapter 4 as the interactive editing engine.

The remainder of the chapter is structured as follows. In Section 5.1, we discuss how sound free-choice workflow nets can be used to model the control-flow aspect of process models. In Section 5.2, we provide details for addressing the first subgoal of extracting the information from the event log and presenting it to the user. In Section 5.3, we discuss the second subgoal of *enabling structured and incremental interaction* using the synthesis engine. In Section 5.4, we evaluate our approach in an objective and subjective way. In Section 5.5, we discuss related techniques described in the literature, followed by conclusions in Section 5.6.

5.1 Labeled Free-choice Workflow Nets

As discussed in Chapter 3 and Chapter 4, the class of free-choice workflow nets is used in our approach as it supports broad structural representations and guarantee soundness when used with the synthesis rules. In order to model the control-flow aspect of business processes, the process models should contain activity labels. To address this, we define the so-called labeled free-choice workflow nets.

Definition 38 (Labeled free-choice workflow net)**.** *Let $\mathscr{A}$ be the set of all activity names. A labeled free-choice workflow net $LW = (W, l)$, where $(P, T, F, i, o, \top, \bot)$ is a free-choice workflow net and $l \in T \nrightarrow \mathscr{A}$ is a labeling function which maps a subset of transitions to activities.*

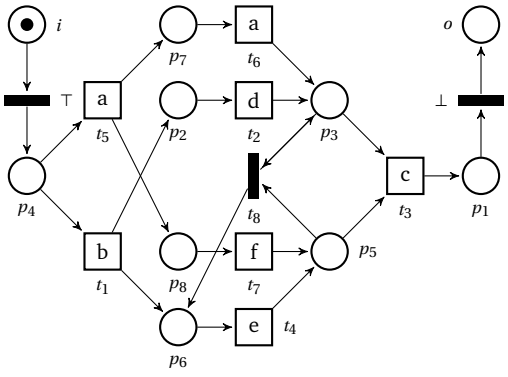
Labeled free-choice workflow nets are essentially free-choice workflow nets, where *some* of the transitions are *labeled* with an activity name. Some transitions may be unlabeled, and represent no behavior (silent activities) in the process. It is also clear that the activities from the labeled free-choice workflow net can be mapped to the activities from the event logs.

Figure 5.2a shows an example of a labeled free-choice workflow net which has some transitions that are labeled with activity names. This labeled free-choice workflow net corresponds to the running example free-choice workflow net from Figure 2.1. Transition t_1 is labeled with activity b, transition t_2 is labeled with activity d, and so on. Transition t_8 is not labeled, i.e., represents a silent transition, and does not exhibit any behavior. It should be noted that, in a labeled free-choice workflow net, multiple transitions can be labeled with the same activity name. For example, in Figure 5.2a, transitions t_5 and t_6 are both labeled with activity a. Figure 5.2b shows an example event log. Every trace from Figure 5.2b can be replayed on the labeled free-choice workflow net in Figure 5.2a.

In the next section, we address the goal of extracting the relevant information from the event log and presenting it to the user in a meaningful way.

5.2 Event Log Information Abstraction and Presentation

In this section, we discuss how the event data can be used to guide the user in modeling a labeled free-choice workflow net. The event logs are central for making decisions in all the automated process discovery techniques. Motivated by the usage of event logs in state-of-the-art process discovery techniques, we derive some statistics from the event logs, which are used to guide the user in decision-making during process model construction. We first discuss how information is extracted from the event log in Section 5.2.1 and Section 5.2.2. Next, we discuss how the extracted information is presented to the user in Section 5.2.3.



(a) An example of a labeled free-choice workflow net.

Trace	Freq
$\langle a, a, f, c \rangle$	8
$\langle a, f, a, e, c \rangle$	1
$\langle a, f, a, e, e, c \rangle$	1
$\langle b, d, e, c \rangle$	4
$\langle b, d, e, e, c \rangle$	1
$\langle b, e, d, e, c \rangle$	4
$\langle a, a, f, e, e, e, c \rangle$	1

(b) An example of an event log.

Figure 5.2: An example of labeled free-choice workflow net and event log.

Table 5.1: Activity-specific information aggregated and presented to the user.

Activity Information	Description
Maximum occurrence	Maximum number of times the selected activity is repeated for any case in the event log
Minimum occurrence	Minimum number of times the selected activity occurs for any case in the event log (can be 0)
Average occurrence	Average number of times the selected activity occurs, averaged over all the cases in which the selected activity happens at least once
% of cases	Percentage of cases in the event log in which the selected activity occurs at least once

5.2.1 Activity-specific Information

From the three synthesis rules, we know that the labeled free-choice workflow net under construction expands one transition (ψ_A^{WF} and ψ_T^{WF}) and/or one place (ψ_A^{WF} and ψ_P^{WF}) at a time. The user labels a newly added transition in the labeled free-choice workflow net with either an activity name from the event log, or the transition does not represent any activity, i.e., leaves the transition silent. Hence, the user can add only one activity at a time while modeling the labeled free-choice workflow net. For a chosen activity to be added to the labeled free-choice workflow net, the user is provided with the aggregated information about the activity as described in Table 5.1.

The activity-specific information from Table 5.1 is useful for gaining insights about the activity such as: is the activity prevalent in the event log?, should the activity be placed in a loop?, should the activity be duplicated?, etc. For example, in the event log from Figure 5.2b, the activity-specific information for activity a is shown in Table 5.2. By looking at the information presented, it is clear that activity a is skipped in 45% of the cases. Furthermore, whenever a happens in a case it always happens twice, providing a strong indication that a must be duplicated, instead of placing it in a loop.

Table 5.2: Information regarding activity a from Figure 5.2b

Activity Information	Value
Maximum occurrence	2
Minimum occurrence	0
Average occurrence	2
% of cases	55%

5.2.2 Contextual Information for Selected Activity

Activity-specific information is useful to analyze the frequency-oriented information about an activity. However, it does not provide indicators about the context in which the activity occurs and hence *where* to place an activity in the process model. In order to assist the user in effectively modeling a labeled free-choice workflow net, we

Table 5.3: The co-occurrence values for the example event log from Figure 5.2b.

$C(\downarrow, \rightarrow)$	a	b	c	d	e	f
a	1	0	1	0	3/11	1
b	0	1	1	1	1	0
c	11/20	9/20	1	9/20	3/5	11/20
d	0	1	1	1	1	0
e	1/4	3/4	1	3/4	1	1/4
f	1	0	1	0	3/11	1

also provide contextual information about a selected activity with respect to the other activities that are already present in the labeled free-choice workflow net.

The information from the event log is aggregated in a pairwise manner between the activity selected by the user to be added to the net and other activities from the labeled free-choice workflow net, i.e., the activities represented by the visible transitions. This assists the user in positioning the selected activity in the net. The pairwise information could be presented in a tabular form to the user, or could be projected directly on the current labeled free-choice workflow net.

The first kind of statistic that we define is the co-occurs value between two activities. This is useful in order to identify activities which (do not) occur together, which is useful in circumstances when a user has to add a certain activity as a choice with respect to some other pre exiting activities in the labeled free-choice workflow net.

Definition 39 (Co-occurs)**.** *Let $LW = (W, l)$ be a labeled free-choice workflow net, such that $W = (P, T, F, i, o, \top, \bot)$ and $l \in T \not\rightarrow \mathscr{A}$. Let L be an event log corresponding to LW. Consider two activities $a, b \in \mathscr{A}$. The co-occurs value of (a, b), denoted by $C_L(a, b)$, is:*

$$C_L(a,b) = \frac{|[\sigma \in L \mid a \in \sigma \wedge b \in \sigma]|}{|[\sigma \in L \mid a \in \sigma]|}$$

For a pair of activities a, b, if the co-occurs value is 0, then a and b do not occur together, whereas if the co-occurs value is 1 then b occurs if a occurs. It should be noted that, the co-occurs value is not commutative. This is because, the co-occurs value is calculated using the absolute occurrence of first activity in the denominator. Table 5.3 shows the co-occurs value corresponding to the event log from Figure 5.2b.

Next, we define the eventually follows value, which indicates the number of times an activity is eventually followed by another activity. This value indicates where an activity should be positioned, compared to all the activities already present in the labeled free-choice workflow net. Note that the co-occurs value only indicates if two activities co-occur together, whereas an eventually follows value indicates whether, and to what extent, an activity is eventually followed (preceded) by another activity.

Definition 40 (Eventually follows)**.** *Let $LW = (W, l)$ be a labeled free-choice workflow net, such that $W = (P, T, F, i, o, \top, \bot)$ and $l \in T \not\rightarrow \mathscr{A}$. Let L be an event log corresponding to LW. Consider an activity $a \in \mathscr{A}$.*

Table 5.4: The eventually follows values for the trace $\langle b, e, d, e, c \rangle$ in Figure 5.2b, assuming all the activities are present in the labeled free-choice workflow net.

$EF_{\langle a,a,f,e,e,e,c \rangle}(\downarrow, \rightarrow)$	a	b	c	d	e	f
a	–	0	0	0	0	0
b	0	–	1	1	1	0
c	0	0	–	0	0	0
d	0	0	1	–	1/2	0
e	0	0	1	1/2	–	0
f	0	0	0	0	0	–

Then, for a trace $\sigma \in L$, and an activity $b \in \mathscr{A}$ $(b \neq a)$, let $\#a >_\sigma b$ indicate the number of occurrences of b after the first occurrence of a in σ. The eventually follows relationship between a and b for a trace σ, denoted $EF_\sigma(a, b)$, is:

$$EF_\sigma(a,b) = \begin{cases} \dfrac{\#a >_\sigma b}{\#a >_\sigma b + \#b >_\sigma a} & \text{, if } \#a >_\sigma b \neq 0 \\ \\ 0 & \text{, otherwise.} \end{cases}$$

The eventually follows relationship between a and b for the entire event log L, denoted $EF_L(a, b)$, is defined as:

$$EF_L(a,b) = \begin{cases} \dfrac{\sum_{\sigma \in L} EF_\sigma(a,b)}{|[\sigma \in L \mid a \in \sigma \wedge b \in \sigma]|} & \text{, if } |[\sigma \in L \mid a \in \sigma \wedge b \in \sigma]| > 0 \\ \\ 0 & \text{, otherwise.} \end{cases}$$

If $EF_L(a, b) = 1$, then a and b co-occur, but a never occurs after a b, which hints that b should (eventually) be following a. Table 5.4 shows eventually follows values corresponding to one of the traces from Figure 5.2b. Similarly, *eventually precedes* value is also calculated, such that the sum of eventually follows and precedes values for a pair is either 1 or 0.

The next statistic derived is the directly follows relation between a pair of activities. This statistic only considers the directly following/preceding occurrences of an activity with respect to the activities present in the labeled free-choice workflow net. The *directly follows* relation for two activities a and b is calculated as follows:

Definition 41 (Directly follows ($DF_{(X,a,b)}$))**.** *Let $LW = (W, l)$ be a labeled free-choice workflow net, such that $W = (P, T, F, i, o, \top, \bot)$ and $l \in T \not\rightarrow \mathscr{A}$. Let L be an event log and let $X \subseteq \mathscr{A}$, such that $X = \{l(t) \mid t \in dom(l)\}$. Consider an activity a, let $L' = L \downarrow_{X \cup \{a\}}$, where $L \downarrow_{X \cup \{a\}}$ is the projected event log such that every trace from L' is projected on $X \cup \{a\}$. Then, for a trace $\sigma \in L'$, and an activity $b \in X$ $(b \neq a)$, let $\#a >^d_\sigma b$ indicate the number of times a is directly followed by b in σ. The directly follows relationship between a and b*

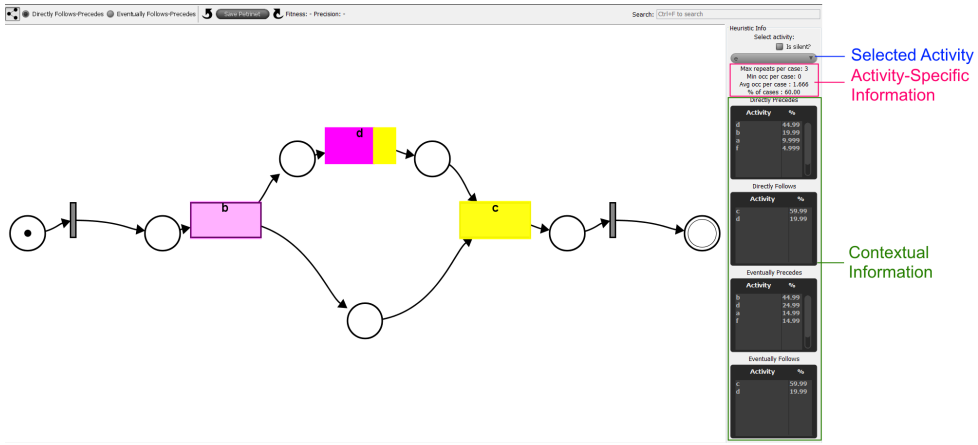


Figure 5.3: Screen-shot of our tool, showing how the abstracted information is presented to the user, based on the selected activity.

.

for a trace σ, denoted $DF_\sigma(X,a,b)$, is:

$$DF_\sigma(X,a,b) = \begin{cases} \dfrac{\#a >^d_\sigma b}{\#a >^d_\sigma b + \#b >^d_\sigma a} & \text{, if } \#a >^d_\sigma b \neq 0 \\ 0 & \text{, otherwise.} \end{cases}$$

The directly follows relationship between a and b for the entire event log L, denoted $DF_L(X,a,b)$, is defined as:

$$DF_L(X,a,b) = \begin{cases} \dfrac{\sum_{\sigma \in L'} DF_\sigma(X,a,b)}{|[\sigma \in L' \mid a \in \sigma \wedge b \in \sigma]|} & \text{, if } |[\sigma \in L' \mid a \in \sigma \wedge b \in \sigma]| > 0 \\ 0 & \text{, otherwise.} \end{cases}$$

We can compute directly precedes relation, such that the sum of directly follows and precedes values for a pair is either 1 or 0. Consider a sequence $\langle b,e,d,e,c\rangle$ in Figure 5.2b. Let activities b,d and c already be added in the labeled free-choice workflow net (as shown in Figure 5.4). Then the directly follows values corresponding to e according to Definition 41 are as follows: $DF_{\langle b,e,d,e,c\rangle}(e,b) = 0$, $DF_{\langle b,e,d,e,c\rangle}(e,d) = 0.5$, $DF_{\langle b,e,d,e,c\rangle}(e,c) = 1$, $DF_{\langle b,e,d,e,c\rangle}(b,e) = 1$, $DF_{\langle b,e,d,e,c\rangle}(d,e) = 0.5$, $DF_{\langle b,e,d,e,c\rangle}(c,e) = 0$.

We now discuss how the extracted information is presented to the user.

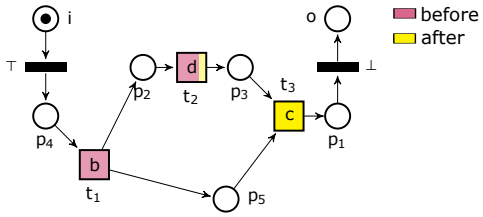
5.2.3 Event Log Information Presentation

In this section, we discuss how the information from the event log is presented to the user. Figure 5.3 shows the screenshot of our tool. The abstracted information

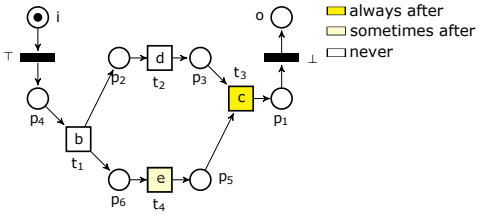
calculated in the above sections is presented in the *raw* format to the user based on the selected activity. Whenever a user selects an activity, the activity-specific information is presented below the name of the selected activity. Furthermore, for the contextual information, the pair-wise values between the activity selected and the activities from the net are converted to percentages and presented in the form of tables for each of the metrics discussed in Section 5.2.2.

The contextual information is also projected on the net, in order to visually assist the user in decision-making. We now discuss the strategy used for projecting the information from the event log on the labeled free-choice workflow net. As a first step, the user selects an activity from the event log that should be added to the labeled free-choice workflow net. Depending on the activity selected by the user, the coloring of the activities currently present in the labeled free-choice workflow net is updated. The colors indicate which activities (and to what extent) from the labeled free-choice workflow net occur before and/or after the selected activity. The projected information can be based either on the eventually follows (precedes) relation, or the directly follows (precedes) relation as desired by the user. The opacity of the colors indicate the co-occurrence of the two activities. As discussed above, all the information from the event logs is also presented in a tabular format to the user. Thereby, in situations where the projected visualizations seem vague, the user can use the information from the tables for making an informed decision.

Figure 5.4a shows the projection on transitions when an activity e is selected by the user. The degree of *purple (yellow)* color in a transition indicates that the selected activity occurs *after (before)* the activity represented by the transition. As transition t_1 is completely colored purple, we know that activity e occurs after activity b. Likewise,



(a) Activity selected: e. The pairwise relations between activities are: $EF_L(b,e) = 1, EF_L(e,d) = 2/9, EF_L(e,c) = 1, C_{(e,d)} = 3/4, C_{(e,b)} = 3/4$ and $C_{(e,c)} = 1$. Upon adding e and selecting e again, we would get the exact same projections. Moreover, e would be colored white, as e happens only once per case in the event log of Figure 5.2b.



(b) Activity selected: a. The pairwise relations between activities are: $EF_L(a,c) = 1$, $EF_L(a,e) = 1$, $C_{(a,b)} = 0$, $C_{(a,d)} = 0$, $C_{(a,c)} = 1$ and $C_{(a,e)} = 3/11$. Activity c is colored dark yellow and activity e is colored light yellow, based on the co-occurrence values. The net results in a similar projection when the activity selected is f.

Figure 5.4: Eventually follows/precedes projections on Figure 5.2b. The purple (yellow) color indicates the degree to which selected activity occurs after (before) the activity represented by the transition.

as transition t_3 is completely colored yellow, we know that activity e occurs before activity c. In contrast, as transition t_2 is colored both purple and yellow we know that activity e occurs both before and after activity d (more often after d). The opacity of the coloring indicates the co-occurrence values of the activity chosen and the activities represented by the transitions. Based on these insights, it is clear that activity e must be added in parallel to d, before c and after b, i.e., by using the abstraction rule (ψ_A^{WF}) between $\{t_1\}$ and $\{p_5\}$.

Figure 5.4b shows the projection on the net when activity a is selected. If a transition is colored white, it implies that the activity selected and the activity represented by the transition never co-occur together. Since transitions t_1 and t_2 are colored completely white, we can say that whenever the selected activity a occurs neither activity b nor d occurs. Multiple transitions having the same label would all have the same coloring. Hence, if both the a's are added to the process model as shown in Figure 5.2a, they would both have identical coloring, depending on the activity selected. Since both t_3 and t_4 are colored yellow, we know that the selected activity a always happens before the activities c and e. Moreover, since transition t_3 is colored darker than transition t_5, we know that activity a co-occurs more often with activity c than with activity e.

Having discussed the projection of statistics on the labeled free-choice workflow net, we now discuss the interaction capabilities of our tool. That is, the way in which synthesis rules are used based on the proposed synthesis engine, in order to enable interactive process modeling.

5.3 Interaction using the Synthesis Engine

Following the projection of the information from the event log, we now focus on the actual user interaction in order to edit and build a labeled free-choice workflow net, using the synthesis engine proposed in Chapter 4. At any given point, the user is not provided with *all* the candidates from the synthesis space explicitly. Doing so would be overwhelming for the user as the number of candidates in the synthesis space rises quickly when the size of the net increases. Instead, we make use of the fact that the candidates from the synthesis space can be tracked back to the corresponding synthesis rule. The user interacts with a labeled free-choice workflow net, in order to deduce new nets, by applying one of the synthesis rules, and thus generating the corresponding candidate net from the synthesis space. This synthesis rule based interaction guarantees that all the labeled free-choice workflow nets from the synthesis space are implicitly available to the user. The user *chooses* a single application of one of the rules, in order to synthesize a net. We begin by discussing the interactive application of the ψ_A^{WF} rule followed by the interactive application of ψ_P^{WF} and ψ_T^{WF} rules.

5.3.1 Interactive Application of the ψ_A^{WF} rule

The usage of abstraction rule is rather straightforward. In order to use the ψ_A^{WF} rule, the user clicks on a (set of) arc(s), and presses *enter*. The selected arcs are highlighted

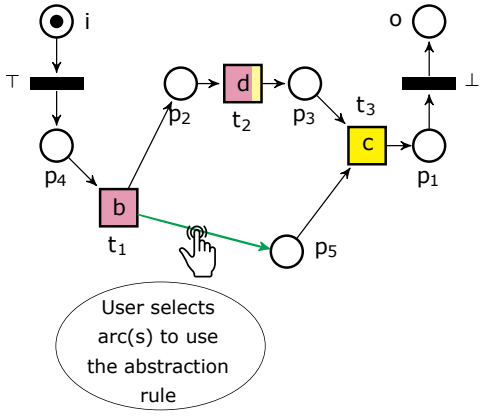
in green. The abstraction rule allows the addition of a new place and a new transition in between a set of transitions and a set of places. The (optional) activity label of the new transition is preselected by the user, after which the rule is applied. Depending on the label chosen, appropriate colors of the transitions in the net are populated as discussed in Section 5.2. Since this rule can be locally checked, a new place and a new transition are added to the net only if the user has selected the valid arcs, that result in the correct application of the ψ_A^{WF}.

Figure 5.5 shows the interaction mechanism behind theψ_A^{WF} rule. We now discuss the interaction mechanism for the ψ_P^{WF} and ψ_T^{WF} rules.

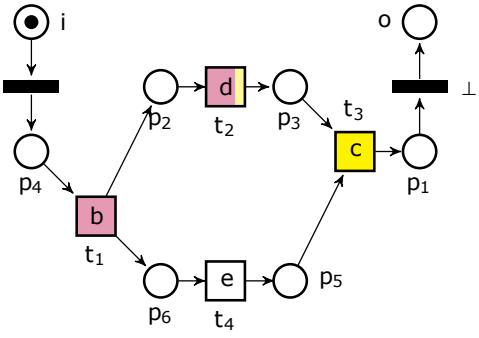
5.3.2 Interactive Application of the ψ_P^{WF} and ψ_T^{WF} Rules

In the case of linear dependency rules, all the possible applications of a rule are projected on the labeled free-choice workflow net based on the user interaction. We know that the ψ_T^{WF} (ψ_P^{WF}) rule allows addition of a linearly dependent transition (place) to a labeled free-choice workflow net. We first explain the central idea that enables interaction based on the synthesis space using the ψ_T^{WF} rule, followed by a couple of examples.

All the candidate applications of the ψ_T^{WF} are precomputed using the incremental synthesis structure based approach. This set of candidate transitions is finite, and independent of the activity label chosen by the user. Whenever a user navigates on a



(a) Selecting an arc to use the ψ_A^{WF} rule. Upon clicking on an arc, it turns green, if it is possible to use the ψ_A^{WF} rule. The ψ_A^{WF} rule is applied after the user has selected the desired arcs and presses enter.

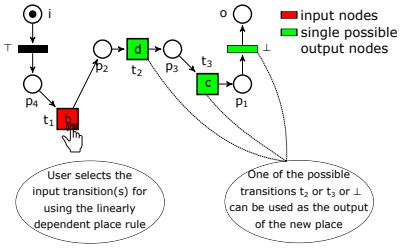


(b) A new transition t_4 and a new place p_6 are added between $\{t_1\}$ and $\{p_5\}$ using the ψ_A^{WF} rule in 5.5a.

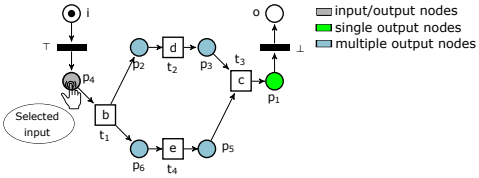
Figure 5.5: Interactively adding activity e using the ψ_A^{WF} rule.

.

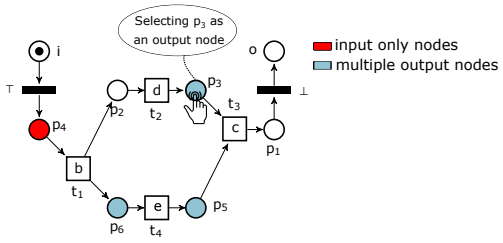
place, all the candidate transitions are explored and only those candidate transitions are presented which have the navigated place as an input place. For all such candidate transitions, the output places (and possible additional input places) are projected on



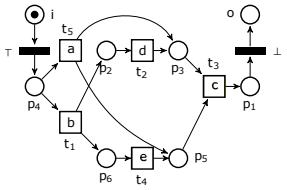
(a) Interactive application of ψ_P^{WF} rule. Upon navigating on transition t_1, all the possible applications of the ψ_P^{WF} rule are projected which have t_1 in its input. The user can select t_1 as the input node which is colored red, and select *one* of the possible output nodes colored in green. Upon selecting t_1 as the input and t_3 as the output, a new place is added, resulting in a net shown in Figure 5.5a.



(b) Demonstration of the ψ_T^{WF} rule. Possible outputs are shown when place p_4 is navigated upon or selected as the input. Dark gray colored *input* place indicates the possibility of a *self-loop*, i.e., adding a transition having same input and output place(s). Blue colored places indicate candidate transitions with multiple outputs, for the selected input place. Each green colored place indicates candidates with a single output place for the selected input place(s).



(c) After selecting place p_3 as an output place, the input place (i.e., place p_4) turns red, indicating that it is no longer possible to use this place to add a *self-loop* transition. The green place p_1 is filtered out, as such candidate transitions are no longer valid for the selected input-output place combination. Moreover, place p_2 is also colored white as there is no candidate transition which contains both the place p_2 and the place p_3 in its output places.



(d) After selecting place p_5 as another output, an application of ψ_T^{WF} could be realized. Hence, a new transition t_5 is added to the net with an input place p_4 and output places p_3 and p_5. The newly added transition is labeled with the preselected activity name a.

Figure 5.6: User interaction for the ψ_P^{WF} and ψ_T^{WF} rules.

the labeled free-choice workflow net, using color coding. The user first selects the desired input places of the candidate transition. The candidate transitions are filtered based on the selected input places, and the output places of the filtered candidate transitions are highlighted. Next, the user chooses the output places. In case of multiple output places for a candidate transition, the user clicks on one of the desired multiple output places. The candidate transitions are further filtered based on the output places chosen. When a user has selected enough input and output places to pinpoint a single candidate transition, the selected candidate transition is added to the labeled free-choice workflow net. In case of multiple input/output places, the number of possible options are eliminated depending on the user's selection. The newly added transition is labeled with the preselected activity by the user. A similar approach is followed for the ψ_P^{WF} rule.

In order to demonstrate the usage of ψ_P^{WF} rule, we use Figure 5.6a. Since ψ_P^{WF} introduces a new place in the net, selection of activity is not required. When transition t_1 is navigated upon, the net is projected with the possible applications of ψ_P^{WF} rules, having t_1 as the input. Since the transitions t_2, t_3 and $\perp$ are all colored green, it is possible to add a new place using either of the three transitions as the output.

Figure 5.6b demonstrates the interactive application of ψ_T^{WF} rule. Assume that the user has selected activity a to be added to the labeled free-choice workflow net. For the sake of convenience, we do not show the event log projections discussed in Section 5.2.3. Upon selecting p_4 as the input and p_3 as one of the outputs, some of the prior candidate transitions are filtered out, as evident in Figure 5.6c. Selecting p_5 as another output pinpoints to a candidate transition, and hence a new linearly dependent transition t_5 is added to the net as shown in Figure 5.6d.

5.4 Implementation and Evaluation

The proposed approach is supported by the nightly build version of ProM [171], in the *InteractiveProcessMiningLite 6.9.13* package, under the plug-in "Petri net editor with log heuristics information". This plug-in requires an event log in XES [3] format as its input.

In order to evaluate our approach of interactive process modeling, we present two types of evaluations: (i) an objective comparison with traditional process discovery approaches, and (ii) a subjective validation via a case study using a dataset from a Dutch hospital.

5.4.1 Objective Evaluation : Process Discovery Contest

As a part of an objective evaluation of the proposed approach, we use our winning entry from the annual process discovery contest [2], organized at the BPI workshop at the Business Process Management conference 2017. The aim of the process discovery contest is to evaluate tools and techniques which are able to discover process models from incomplete, infrequent event logs. In total, there were 10 process models that had to be discovered from 10 corresponding event logs. Every model could contain

Table 5.5: Process models and their corresponding optional characteristics.

Model	Characteristic 1	Characteristic 2	Incomplete event log
2	Optional activities	Loops	Yes
7	Duplicate activities	Inclusive Choices	No
10	Optional activities	Duplicate activities	Yes

sequences, choices and concurrent activities. Furthermore, the organizers provided participants with some additional information about each process model, such as the presence of duplication, loops, skippable (optional) activities, inclusive choices and long-term dependencies. We considered this additional information as the *domain knowledge* in our approach. In order to improve the accuracy of discovered process models, the organizers provided two feedback loops of small test event logs to be evaluated by the participants on their models. The process models were evaluated based on the number of correctly classified traces. That is, the traces which completely fit the process model v/s the traces that do not fit the model. Furthermore, the process models were also evaluated by a jury composed of BPM practitioners for simplicity and understandability.

In this section, we present the outcomes of three event logs (2, 7 and 10) for discovering process models using our approach, as well as state-of-the-art process discovery approaches. The characteristics of the three selected models are shown in Table 5.5. The information from Table 5.5 is used along with *directly follows projections*, *eventually follows projections* and other extracted information from the event

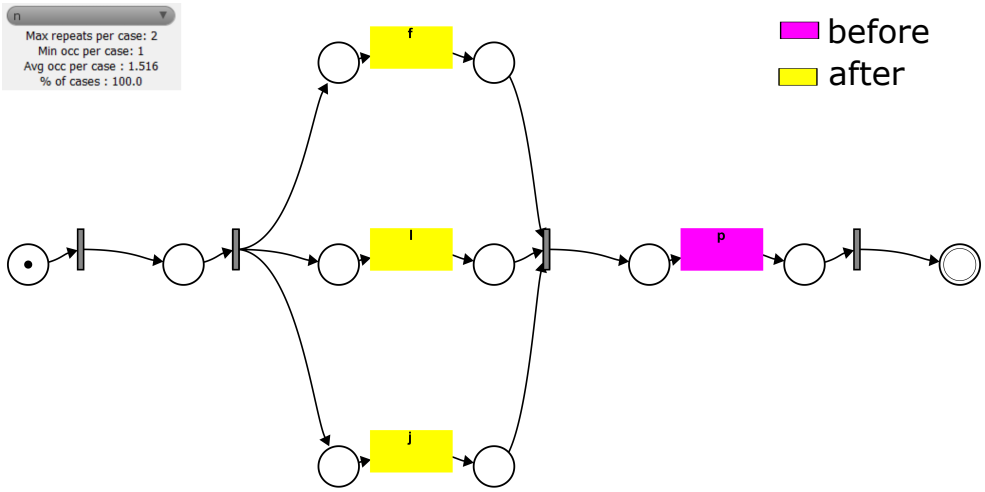


Figure 5.7: Model 7. Activity selected: *n*. The projections indicate directly follows relations between all the activities in the net and the chosen activity *n*.

logs, to construct the process models. Figure 5.7 - Figure 5.12 show the screenshots of some of the steps followed in order to discover the process models using our approach.

Figure 5.7 demonstrates a way in which duplicate activities were identified and added to the process model corresponding to event log 7 using our approach based on directly follows/precedes relation. From the activity specific information we knew that n could happen a maximum of two times for any case. Furthermore, according to Table 5.5, model 7 does not contain any loops. Hence, n must be duplicated and added twice, once before activities f, l and j, and once after activity p, as indicated by the projections on the net.

Figure 5.8 shows a way in which a choice constraint was added to the net. The projections indicate eventually follows relations between all the activities in the labeled free-choice workflow net and the chosen activity h. Clearly h and e never co-occur together - as evident by the white coloring of activity e. Moreover, h occurs (eventually) after all the other activities from the labeled free-choice workflow net. Hence h is added as a choice to activity e. The result after adding h (and j and i) is shown in Figure 5.9.

Figure 5.9 also shows the way in which concurrency was identified and introduced in a net. For the selected activity k, and eventually follows/precedes relations, activity d is colored fifty-fifty, indicating that k happens sometimes before, and sometimes after d. Moreover, the coloring of other activities indicate that k should happen concurrently to d. Hence, we first added a place using the ψ_P^{WF} rule with input as the transition labeled a and the output as the transition labeled b and the silent transition (in choice with b). Next, k was added using the ψ_A^{WF} rule between a and the newly added place, thereby in parallel to b. Figure 5.10 shows the net obtained after using both ψ_P^{WF} and ψ_A^{WF} to introduce k.

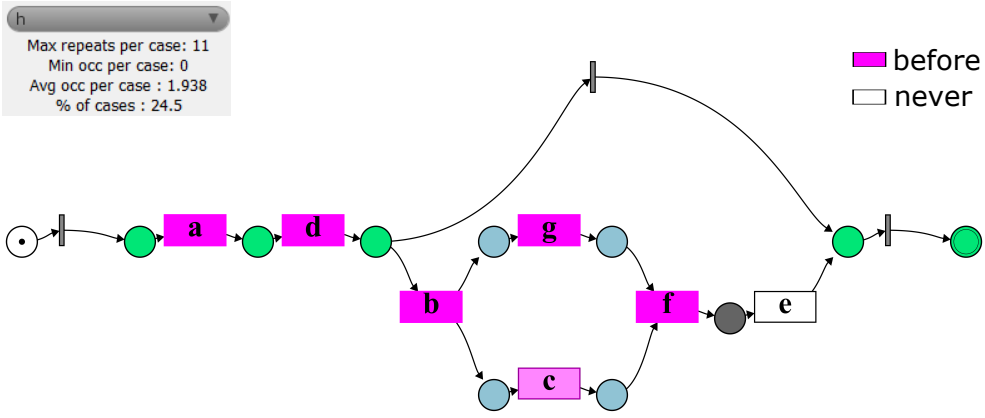


Figure 5.8: Model 2. Activity selected: h. Eventually follows relation indicates indicate that h occurs in choice with e.

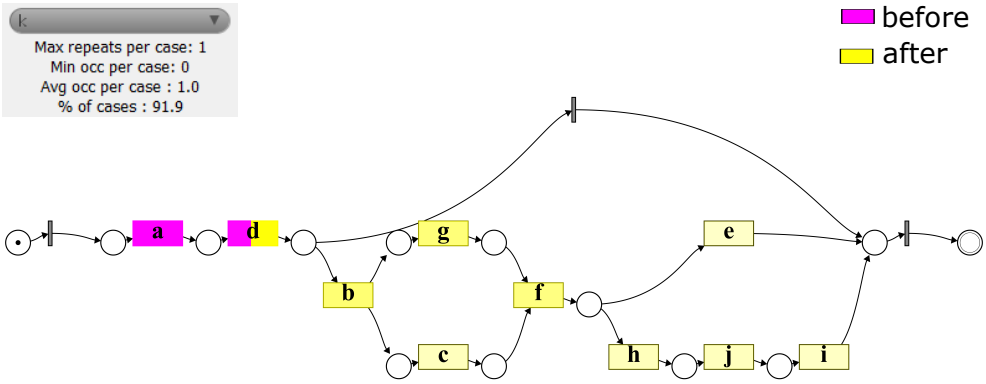


Figure 5.9: Model 2: Selecting activity k to be added to the labeled free-choice workflow net.

Figure 5.10 also shows the way in which sequential construct was identified and added to the net. The eventually follows/precedes relation indicates that the selected activity m happens after a but before all the other activities. Hence m was added in sequence using the ψ_A^{WF} rule based on the selected green arcs. The resulting net is shown in Figure 5.11, also containing the activity l.

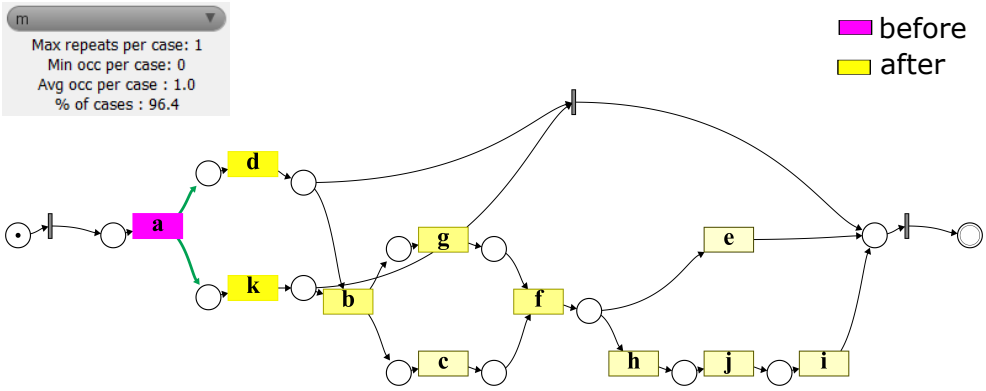


Figure 5.10: Model 2. Activity selected: m.

Figure 5.11 also shows one of the limitations of using the synthesis rules for constructing process models. Clearly the selected activity o should happen after *both* a and n, but before l (and all other activities). However, no rule allows this. Note that the ψ_A^{WF} cannot be used here, as the output places of a and n are not the same. A

workaround for this is to first re-name the transition labeled l with the label o. This is possible as any transition could be (re-)labeled with any activity in our approach. After re-naming l with o as shown in Figure 5.12, we could re-add activity l (after o using the ψ_A^{WF} rule) to the net.

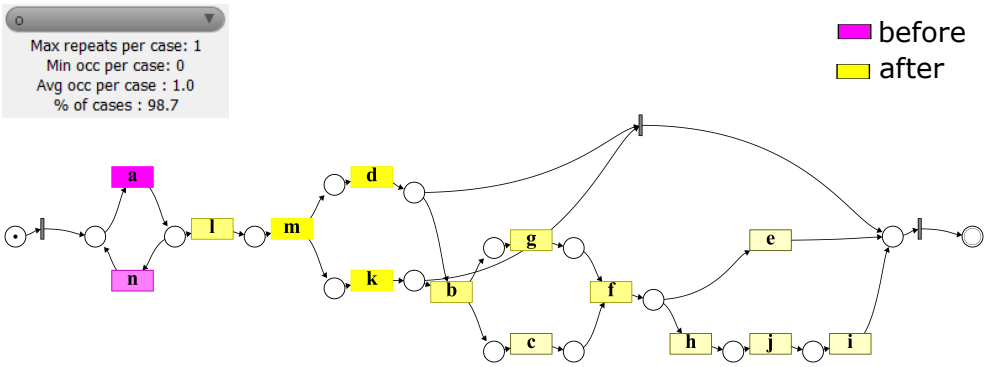


Figure 5.11: Model 2. Activity selected o. No rule allows adding o after both a and n, but before l.

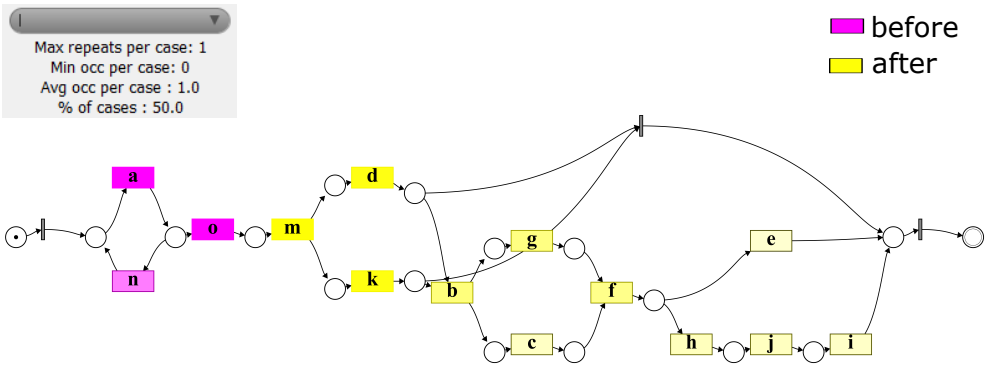
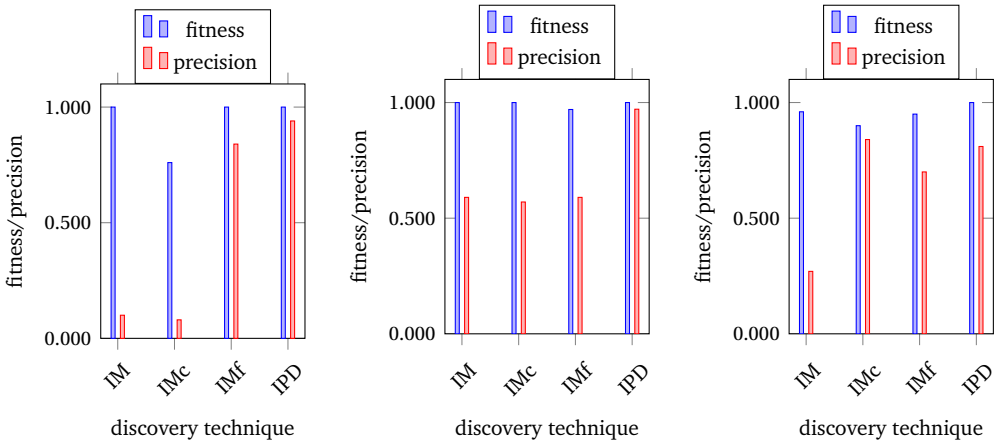


Figure 5.12: Model 2. Activity selected: l, which can now be re-added after o and before m.

A similar approach was followed for deducing the remaining process models. In some scenarios, when the projected and tabular data did not portray sufficient information for decision making, simple log visualizers presented in [171] were also used to assist decision making.



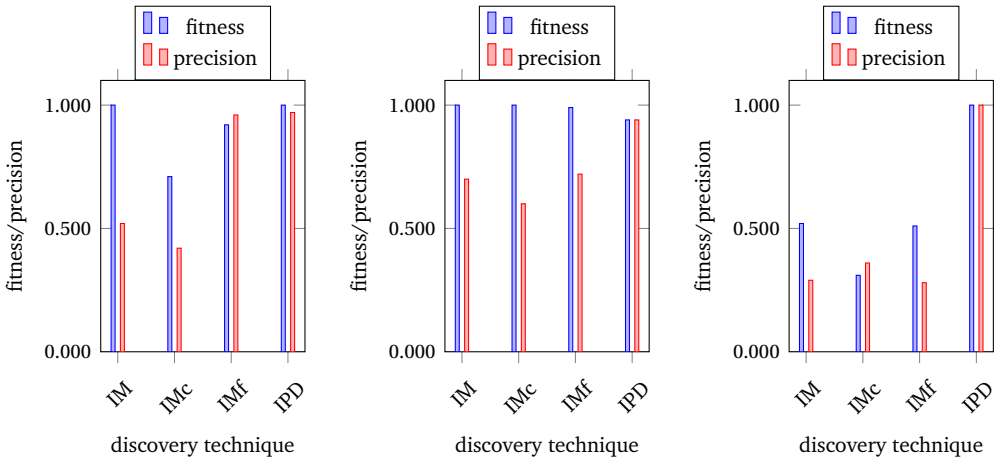
(a) Model 2 (b) Model 7 (c) Model 10

Figure 5.13: Fitness and precision scores computed using the alignment based conformance analysis [5, 128] on the training event log and models. (IM: Inductive Miner, IMc: Inductive Miner incompleteness, IMf: Inductive Miner infrequent, IPD: Interactive Process Discovery - proposed technique).

All the three process models (2, 7 and 10) had *100% accuracy* in classification during both the intermediary test event logs used for the feedback, as well as the test event logs used for the final evaluation.

In Figure 5.13 we compare the fitness and the precision scores of the training event log used to discover a process model, with the process model discovered, using alignment based conformance technique [5, 128]. We do not consider other automated approaches such as the ILP miner [170] or the Heuristics miner [185] as they could discover process models which are unsound and/or unbounded, for which metrics such as fitness and precision cannot be computed. Both the fitness and precision values are calculated on a scale from 0-to-1. Fitness score indicates how well an event log fits a process model. Inductive miner by default guarantees perfect fitness. Hence, in all three models discovered by the Inductive miner, the fitness score is a perfect 1. Other variants of inductive miner also have a respectable fitness score, above 0.8, in all three cases. The precision score indicates how much extra behavior is allowed by a process model, that is not observed in the event log. Clearly, the inductive miner variants struggle with producing precise models, and end up discovering process models which are too generic. On the contrary, the process models discovered using the interactive approach have high fitness as well as high precision scores.

The original process models (so-called *system* nets) which were used to generate the event logs were also made available at the end of the competition. We also used these original process models as a benchmark, and compared it with the process models discovered using various discovery techniques, including our technique. In order to compare two models, we used the Projected Conformance Checking (PCC) techni-



(a) Model 2 (b) Model 7 (c) Model 10

Figure 5.14: Comparison of the original model used to generate the event log with the discovered models. The comparison is done in terms of fitness and precision values using the PCC framework [112]. (IM : Inductive Miner, IMc: Inductive Miner incompleteness, IMf: Inductive Miner infrequent, IPD: Interactive Process Discovery - this paper).

que [112]. Exploring the complete state space would of course be time consuming, and infeasible with limited resources. Instead of comparing the complete state space of the two models, the PCC framework approximates the recall (also known as fitness) and precision values by comparing the state space of smaller process models and then aggregates the scores. We used a value of $k=2$, to project a process model on a subset of 2 activities iteratively in order to compute the recall and precision scores.

The recall value indicates the part of the behavior of the original process model that is captured by the discovered process model. The precision value on the other hand captures the part of the behavior of the discovered process model that is also present in the original process model. The results for these are shown in Figure 5.14.

As evident from Figure 5.13 and Figure 5.14, our approach can outperform the automated process discovery techniques. By using the characteristics of the processes as *prior knowledge*, we were able to interactively discover process models which were strikingly similar to the original process models. Moreover, we could easily convert the discovered process models into BPMN format which were simple and easy to understand as judged by the jury of the contest. As an example for demonstrating the simplicity of the discovered process models, Figure 5.15 shows the process model for model 10 discovered using our approach.

All three process models discovered were able to replicate all the behavior from the test event logs. Furthermore, overall the 10 models, we achieved an accuracy of 98.5% with respect to the test event logs, i.e., of the 200 test traces we had to classify, we classified 197 correctly.

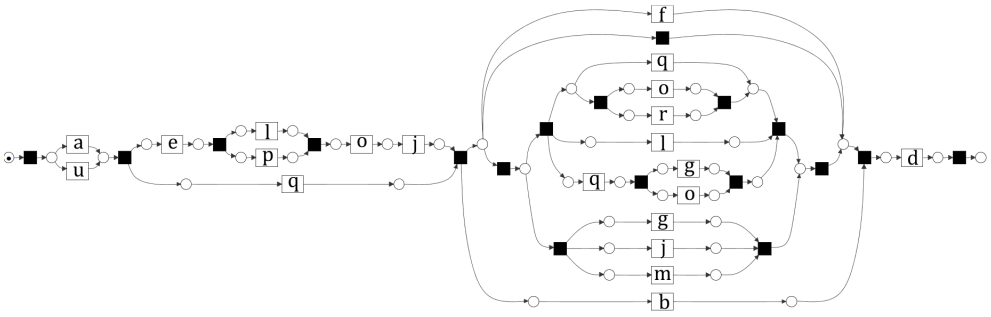


Figure 5.15: Model 10 of the process discovery contest discovered using our approach.

5.4.2 Case Study

In this section, we evaluate the usefulness of the proposed approach in a qualitative way. A real-life labeled free-choice workflow net is modeled with the proposed approach by using domain knowledge along with the information from the event log. The case study is performed in collaboration with a local healthcare practitioner who served as the domain expert, by using data on the treatment process of 2 years for a specific type of cancer. The main objectives of the case study were: *(i)* discovery of the overall process; *(ii)* analysis of issues in the execution of the process as depicted by the event log.

Process Discovery Phase

We started off by trying out several automated process discovery techniques using only the event log. Although, in theory, the process should be rather straightforward, the usage of automated process discovery techniques resulted in extremely complex and unreadable spaghetti-like process models or highly imprecise flower-like models, which allowed for any behavior. All the process models discovered by the traditional process discovery techniques were either incomprehensible and/or very far off from reality according to the domain expert. Therefore, the interactive process discovery approach was used to try to structure the process data by using domain knowledge to discover a process model.

The model discovered using our technique is shown in Figure 5.16. As the process structure in the event log was highly ambiguous, we relied on the inputs of the domain experts in order to come to a process model. The domain expert had complete control about the modeling of the process. It is worthwhile to note that, on several occasions, the domain expert also took assistance from insights of the event log, that were presented in the tool. For example, the domain expert was not entirely sure if activity *LvPA* should be placed before or after *MDO*. However, after gaining insights from the data, as projected and shown in Figure 5.17a, the domain expert decided to add *LvPA* before *MDO*. On some occasions, the domain expert chose to ignore the information from the data, deeming it inappropriate and/or inadequate. For example,

as evident in Figure 5.17b, the data presented showed that activity *Pathologie* often happened more than once for several cases. However, as evident in Figure 5.16, the domain expert chose to overrule this information. It should be noted that during the study, the tool had a slightly different look-and-feel, compared to the present version of the tool.

Finally, using the information from the event log and some background knowledge about the process, the domain expert was able to discover, and was content with, a very structured process model.

Comparing Logs and Models Phase

Conformance analysis [5] was performed on the process model discovered using our approach and the event log, to find how well the event log fits the discovered model. The fitness score for the complete event log and the process model was 0.45. The

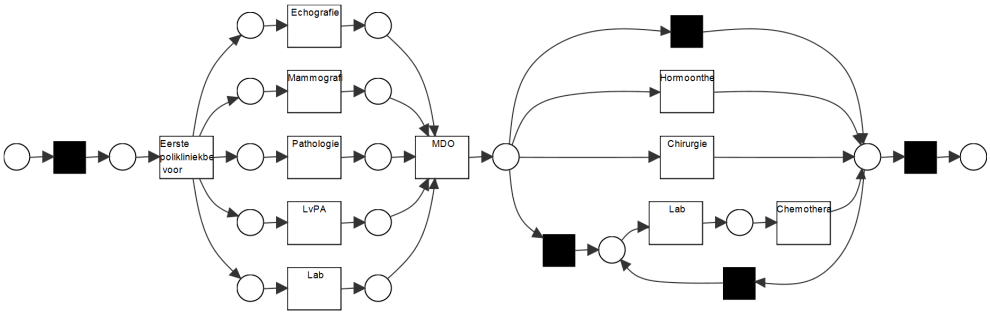
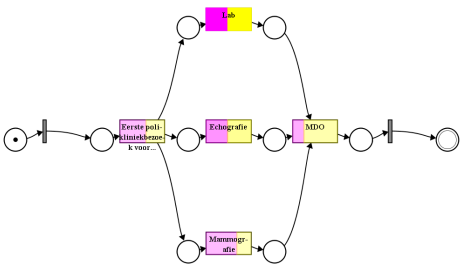
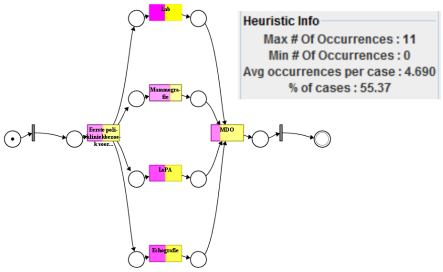


Figure 5.16: Real-life process model for cancer patients in a Dutch hospital as discovered by domain expert using our tool.



(a) Activity selected : *LvPA*. The color code of *MDO* is mostly (around 80%) yellow. This helped the domain expert in positioning *LvPA* before *MDO*.



(b) Activity selected : *Pathologie*. Average number of times *Pathologie* happens for a case is more than 4 times.

Figure 5.17: Screenshots showing some steps in the interactive discovery of the healthcare process.

process model was discovered primarily by the inputs of the domain expert, and is correct according to the domain expert. Hence, a low fitness score of 0.45 indicates at one or both of the following:

- the protocols and the process in place are not strictly followed in the hospital, and/or
- there are data quality issues which lead to incorrect logging of data

Discussion

Upon investigating the causes for the low fitness with the domain expert, the primary conclusion was that there were serious data quality issues in the event log, rather than non-compliance of protocols. There were many events which appeared out of sync with the process and happened sporadically. One of the major causes for this was that many events were recorded to have happened multiple times in the process. Even though, according to the process model, these events should happen only once per case. The probable cause for this issue could be the fact that these events were not extracted correctly, leading to duplication. Another issue, that may have caused many events to appear out of sync in the process could be the fact that many events had no timestamp information associated with them.

Data quality problems are often the reason why all the automated discovery algorithms fail to discover a structured process model. However in our case, by not relying completely on an event log for process discovery, and using knowledge from a domain expert, a structured process model was discovered. Furthermore, it should be noted that even though the process model has a very low fitness score, the model is extremely precise according to the expertise of the domain specialist. Moreover, since the domain expert was directly involved in process modeling/discovery, the simplicity and generalization dimensions of the process model [5], were implicitly taken into account. Clearly, using the interactive process modeling approach led to the discovery of a process model, and brought to the attention of the domain expert various data quality issues which needed to be addressed.

5.5 Related Work

In this section, we compare the relevant techniques from the literature with our approach. We first review the state-of-the-art automated process discovery techniques followed by the user-guided process mining techniques.

5.5.1 Automated Process Discovery

One of the first automated process discovery algorithms proposed was the α miner [169]. The α miner discovers process models, represented by Petri nets, using the directly follows relation in the event log. The original α miner could not discover constructs such as short-loops, long-distance relations, non-free choice constructs

and silent transitions. However, these concerns have been subsequently addressed in [54, 96, 187–190]. These techniques however cannot guarantee discovery of sound process models and are sensitive to noise. The Flexible Heuristics miner [185] and the Split miner [9] can cope with noise well, and can discover non-trivial constructs such as long-distance dependencies. The Flexible Heuristic miner discovers 'causal nets' and the Split miner discovers BPMN models using probabilities based on the directly follows (precedes) and eventually follows (precedes) relations extracted from the event log. These causal nets and BPMN models can then be converted into Petri nets. Fodina [157] extends the original Heuristic miner [186] to support the discovery of constructs such as duplicate activities and long-distance dependencies. Furthermore, Fodina also supports multiple configuration options, and thereby allows the end user to have more control over process discovery. However, Fodina, the Split miner and the Heuristic miner do not guarantee discovery of sound process models.

Approaches based on the state-based region theory, obtain a state space from the event log [166], and then this state space is used to generate a Petri net [33, 46]. However, the computation times of such techniques could be very long, and such techniques do not deal very well with noisy data. Divide-and-conquer strategies based on state-based region theory were proposed in order to address the lengthy computation times [30, 32, 148]. In language-based region theory, a Petri net is discovered which contains the smallest possible behavior while describing the 'language' given by the event log [14, 15]. Approaches such as the ILP miner first add all the activities (transitions) and then add places that do no prohibit any trace from the event log [170]. These approaches typically do not guarantee classic soundness, have difficulties in dealing with noisy data and discovering silent or duplicate activities.

Genetic discovery techniques such as [24] can discover constructs such as silent and duplicate activities, and provide guarantees with regards to soundness. However, such techniques typically have excessive run times and cannot discover non-block structured constructs. The Constructs Competition Miner and the Inductive miner [110] can discover block-structured process models and provide guarantees with regards to soundness of the discovered process model. However, such techniques cannot discover duplicate activities and non-block-structured constructs. Techniques such as [44] focus on discovering complex process models represented as BPMN models containing sub-processes and events, but do not provide any guarantees with regards to the soundness of the discovered process model. Discovery techniques such as [13, 22, 40] discover either constraints based declarative models, or probabilistic models, and not an end-to-end process model.

Our approach differs from all these automated process discovery techniques in multiple ways (see Table 5.6 for an overview). The process models generated by our approach are always sound, since we use the synthesis space based on synthesis rules which guarantees soundness. In our approach, the user has control over the discovery (modeling) process, therefore the addition of constructs such as duplicate activities, silent activities, (self-)loops etc. are all allowed as deemed appropriate by the user. Also, noisy (incomplete) information could be ignored (overcome) based on the extracted information from the event log presented to an informed user. A user who is a domain expert can use the domain knowledge in order to overcome an

Table 5.6: Representational bias of various process discovery and repair algorithms.

	α [169]	HM/SM [9, 185]	ILP [170]	SBR [46]	LBR [14]	ETM [24]	IM [110]	MBR [77]	IPD
Duplicate activities	-	-	-	+	-	+	-	-	+
Silent activities	-	+	-	+	-	+	+	+	+
Self loop activities	-	+	+	+	+	+	+	+	+
Non block structured	+	+	+	+	+	-	-	+	+
Classic soundness*	-	-	+	-	-	+	+	-	+

α : Alpha miner, HM: Heuristic Miner, SM: Split Miner, ILP: ILP miner, SBR: State-based regions, LBR: Language-based regions, ETM: Evolutionary Tree Miner, IM: Inductive Miner, MBR: Model-based Repair, IPD: Interactive Process Discovery (this paper)
Classic soundness* [168] is defined for the class of workflow nets

incomplete event log.

5.5.2 User-guided Approaches

There has been an interest in using domain knowledge, along with the event logs, for process discovery [10, 116, 139]. [90, 91] allow the user to input domain knowledge, in terms of precedence constraints. Causal relations are encoded from the event log and combined with the precedence constraints over the topology of the resulting process models represented as causal nets. However, in such techniques the domain knowledge is represented by some pre-defined constructs or some sort of rules which are used as input during the process discovery. The language used to represent the domain knowledge severely limits the expressiveness of the domain expert. In our approach, the user has total control over the discovery phase and can intuitively use the domain knowledge, along with the event logs, to interactively visualize and discovery a process model. In [122], the authors provide a way for the user to include domain expertise in the α miner. However, this approach is tied to the underlying α algorithm, and thereby includes all the limitations of the α algorithm. Essentially, [122] provides a way of introducing domain knowledge specific to an activity which is then used by the α algorithm, rather than an interactive exploration of the process model. In [98], the authors describe some of the requirements for an interactive workflow mining

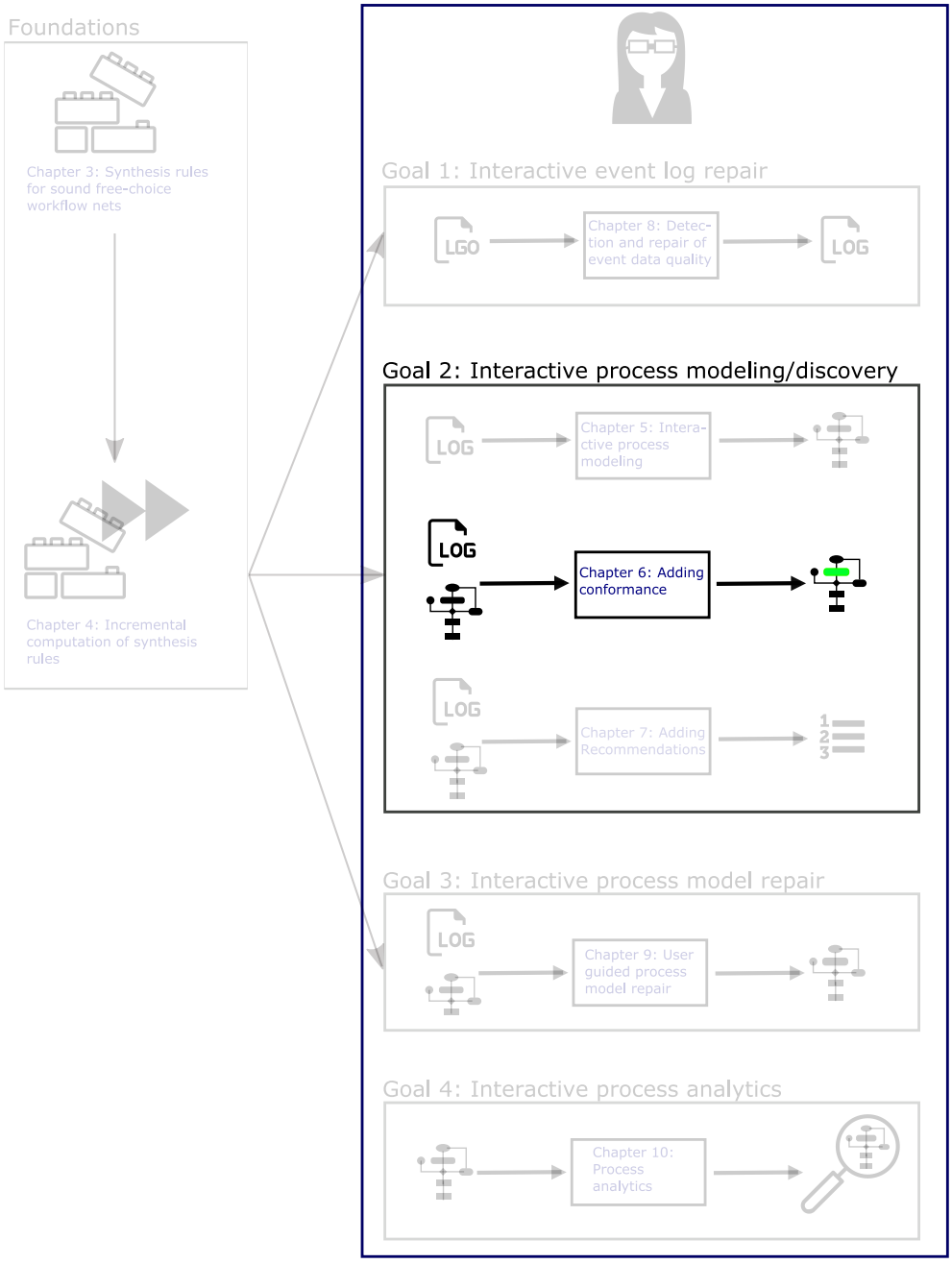
system, and address the needs for layout of workflows to enable interaction as well as provide techniques for preliminary validation of workflow models based on the event log. These techniques have a different scope compared to interactive process discovery as addressed in this chapter. We discuss the (user-driven) process model *repair* techniques from the literature such as [8,78] in the Related Work (Section 9.5) of Chapter 9.

5.6 Conclusion

In this chapter, we presented the concept of interactive modeling of a process model based on the proposed synthesis engine proposed in Chapter 3 and Chapter 4. This approach exploits the fact that the synthesis engine guarantees soundness of the discovered process models. Furthermore, the information from the event log is extracted and presented to the user to assist the user in decision making. Giving users complete control over the discovery approach supported by the information from the event log enables critical decision making. This is true especially when all the information needed by the discovery algorithms is not present in the event log; which is often the case in many real-life event logs as apparent from the case study presented. The automated discovery algorithms fail to cope with insufficient information in the event log, and could produce process models which are incomprehensible and/or inaccurate. Moreover, the proposed approach for interactively modeling process models was able to discover constructs such as non-block-structured representations, duplicate activities, inclusive choices, and silent activities, that cannot be discovered by many state-of-the-art techniques. As a next step, it would be ideal to give the user an indication about the goodness-of-fit between an event log and the corresponding process model, while the user is interactively modeling the process model.

Chapter 6

Fast Conformance Analysis for Interactive Process Modeling



This chapter is based on the publications [62] and [68].

In Chapter 5, we discussed a strategy to enable interactive process modeling using an engine based on the synthesis rules presented in Chapter 3 and Chapter 4. Furthermore, we discussed a way of projecting the information from event logs to assist an analyst in modeling/discovering process models. However, the analyst also needs to be provided with indications about the impact of a change in the process model. That is, the analyst is provided with no indication about the *quality* of the process being modeled. It is desirable to have process models which are correct according to the analyst, but these process models should ideally also have high quality as per the event log.

Typically, the quality of a process model is assessed using the four quality dimensions of process mining (*fitness, precision, generalization and simplicity* [26]). These dimensions could be used as the guiding principle for discovering high quality process models. The *fitness* dimension determines how much behavior from the event log is allowed by a process model. The *precision* dimension, on the other hand, determines how much extra behavior does a process model allow which is not present in the event log. The *generalization* dimension evaluates the degree to which the discovered model is generic, i.e., the extent to which it is coherent with some *unseen* future behavior. Finally, the *simplicity* dimension addresses the comprehensibility of the discovered process model. There is no real consensus on the best way to measure generalization and simplicity, and multiple viewpoints are possible. However, it could be argued that the fitness and precision dimensions could be computed entirely based on the event log and the process model. To calculate the fitness and precision of a

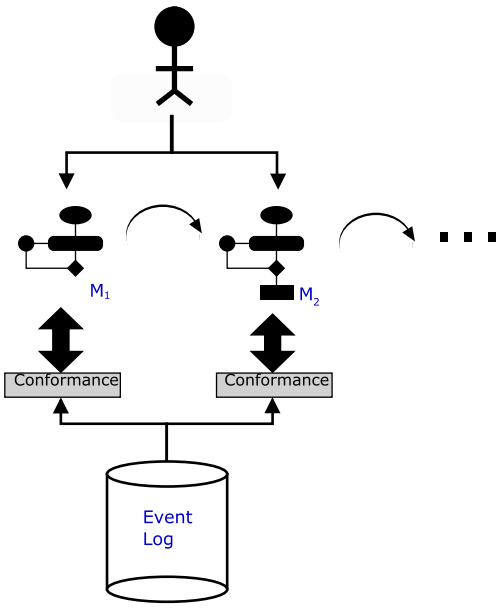


Figure 6.1: Interactive process modeling/discovery enhanced with conformance results.

process model corresponding to an event log, we can use conformance analysis [163]. In an interactive setting, such conformance information could further assist the user in modeling/discovering better process models, by providing a feedback mechanism, as shown in Figure 6.1.

Traditionally, conformance analysis techniques analyze how well a pre-existing process model conforms to the reality as depicted by the event log. Hence, much emphasis is put on the accuracy of the results. These conformance results can then be used to perform analysis such as compliance issues in the process behavior. Furthermore, conformance results can also be used to analyze the performance of the process in reality, for example, to find out where the possible bottlenecks in the process are.

Figure 6.2 shows the times taken for computing conformance scores using some state-of-the-art conformance analysis techniques based on six real-life event logs and the corresponding process models. These publicly available event logs contain broad representations of data. For example, the number of cases vary from 700 to 100,000, whereas the number of events vary from 12,500 to 450,000 across all the event logs. Moreover, the corresponding process models also cover broad representations such as duplicate activities, loops etc. More information about these event logs could be found in the evaluation section (Section 6.4) of this chapter.

Clearly, the time taken to compute conformance scores by the state-of-the-art conformance analysis techniques could be well over 20 seconds, for some of the real-life event logs. As discussed in Chapter 4, it is vital to limit the waiting times suffered by the user to a minimum in an interactive setting. That is, it is undesirable for the user to wait for a long time to get the conformance scores upon making a change in the process model. In the case of interactive process discovery, the in-depth analysis offered by conformance techniques is often not very relevant. Moreover, it can be argued that the accuracy of the conformance analysis may be lower if it is compensated by

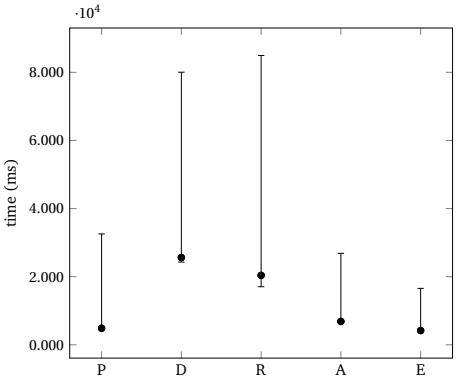


Figure 6.2: Average, minimum, and maximum time taken for computing conformance (fitness and/or precision) using various state-of-the-art techniques (P - PCC framework (k = 2), D - Decomposed Replay, R - Recomposed Replay, A - Alignment-based replay, E - ETC 1-align precision.), based on 6 real-life event logs and their corresponding discovered process models.

improved response times.

Hence, in this chapter, we address the subgoal of computing conformance scores suitable in an interactive process modeling setting, as described in Goal 2 of Chapter 1. In particular, this subgoal is comprised of the following.

- The conformance analysis technique should provide a global and a local, activity specific, measure to (approximately) indicate the fitness and precision of a process model and the event log.
- More importantly, the conformance analysis should be computed in an efficient way, without impacting the interactive capabilities of the system.

Loosely speaking, the complexity of computing conformance scores can be linked to two factors. First, the size and the structural complexity of the process model. The complexity of some conformance analysis algorithms increases upon increase in structural constructs such as loops in the process model. Second, the size of the event log. Typically, the conformance analysis algorithms compute the conformance on a case-by-case basis. Hence an increase in the number of cases and the number of events in the event log could have a detrimental effect in the performance of the algorithm.

In the literature, several techniques have been proposed in order to address the issues related to the complexity and the size of the process model. Typically, such techniques use a divide-and-conquer strategy by dividing the computation of conformance on one large complex net, into several smaller simpler nets [112, 162]. Even though, these techniques address the first factor of process model complexity, they could still suffer from the latter problem related to the size of the event log. These techniques compute conformance of each case in the event log with the process model. Hence, the performance could still be impacted when an event log contains a large number of cases or events. Therefore, these techniques are not directly suitable in an interactive setting.

In this chapter, we address the goal of computing conformance scores in an approximated way, suitable in an interactive setting. We propose strategies to compute conformance scores which address both the factors discussed above. First, we use a divide-and-conquer strategy to split one big conformance problem into several smaller problems, similar to [112]. Next, we also propose a way of abstracting the information from the event log, and using the abstracted information to compute the conformance score, instead of revisiting the event log again and again in an interactive setting. Furthermore, since we deal with the class of free-choice workflow nets synthesized using the synthesis rules, we propose a way to compute the conformance in an incremental way using the underlying nature of the synthesis rules.

The remainder of the chapter is structured as follows. In Section 6.1 we provide an overview of the proposed approach to address the problem of computing conformance in a speedy way. In Section 6.2 and Section 6.3 we provide a particular solution based on the proposed approach. In Section 6.4 we discuss the implementation details and evaluate the proposed solution in terms of speed and accuracy, compared with the state-of-the-art approaches. In Section 6.5, we discuss the techniques from the literature related to our approach, followed by conclusions in Section 6.6.

6.1 Projected and Incremental Conformance Analysis

In order to address the issues pertaining to the usage of conformance analysis in the context of interactive process discovery, we propose a novel incremental approach based on event log abstraction and projections for conformance checking. Before diving into a particular solution, we begin by providing a *generic* approach, which can be instantiated in multiple ways. The approach is centered around two aspects, which are discussed in the following sub-sections.

6.1.1 Project and Aggregate

The central idea behind our approach is motivated by the various divide-and-conquer strategies which were introduced to speed-up conformance calculations. The top part of Figure 6.3 shows a high-level overview of how these techniques typically work [32, 162]. To speed-up the conformance analysis approach, these techniques split up one big conformance problem into several smaller problems. That is, instead of checking the conformance of the complete process model in one go, the process

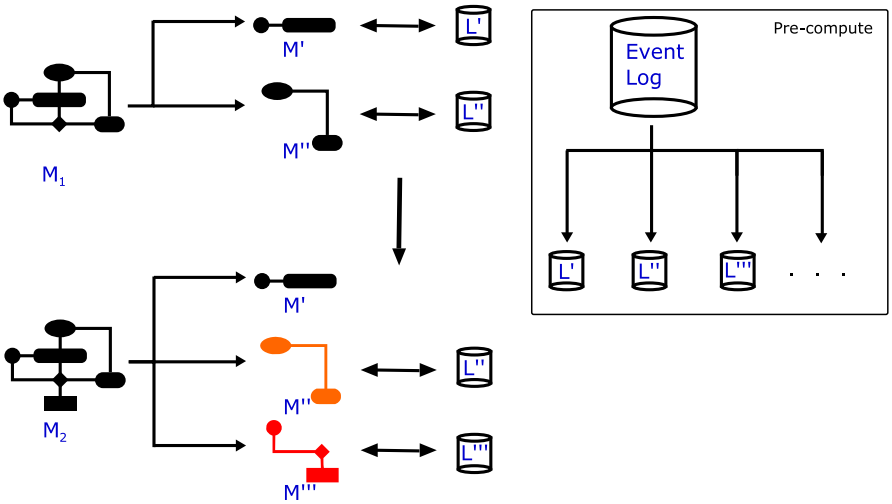


Figure 6.3: The information from the event log is abstracted only once, and used throughout the interactive modeling process. Individual conformance is computed for each of the fragments M', and M'' that together constitute M_1. The individual conformance scores can then be aggregated to get the overall conformance of the Model M_1. Model M_2 is interactively modeled from M_1. The behavior in the fragment of M' is unchanged in M_2, hence there is no need to re-compute the conformance of M'. The behavior of M'' is changed in M_2 compared M_1, whereas M''' is newly introduced in M_2. Hence, the conformance needs to be computed only for M'' and M'''. The aggregated conformance scores of M', M'' and M''' provide the conformance of the interactively modeled Model M_2.

model is *split up* into several smaller fragments. Next, the event log is also split up corresponding to each of these fragments. Then, the fragments from the process model are compared with the smaller event logs in order to calculate the conformance per fragment.

The traditional divide-and-conquer-based conformance techniques can offer sufficient speed-up when conformance needs to be calculated only once. However, in our case, the conformance scores have to be calculated several times when the process model is interactively changed. It should be noted that in an interactive process discovery setting, the event log remains unchanged. Hence, in our approach, instead of splitting the event log into several smaller logs based on each fragment, we propose to pre-compute abstractions on the event log which can be used to compute conformance, as shown in Figure 6.3. We refer to these abstractions as patterns. These patterns are then used to calculate the conformance score by comparing them with similar patterns extracted from process models. The patterns from the event log are computed only *once*, and hence while calculating the conformance, the time spent on processing of the event log is saved. For example, the abstracted part of event log L''' is pre-computed but unused in the top-part of Figure 6.3. However, this comes into effect when the process model M_2 is derived from M_1.

A global measure for quality of a process model compared with the event log can be obtained by aggregating the individual conformance across all the process fragments. Moreover, by aggregating the projections for all the fragments containing a particular activity, we can also get a local activity specific conformance estimate.

6.1.2 Incremental Computation

The second aspect of our approach deals with incrementally computing the conformance scores, based on the changes in the process model. In an interactive process modeling scenario, the changes to a process model are limited in each interactive step. Hence, most of the prior connections from the prior model remain intact. In the second part of the proposed approach, we exploit this fact in order to compute the conformance scores only for those *parts* of the process model which may have changed (or newly introduced). This is shown in the bottom part of Figure 6.3. It can be seen that the decomposed fragment of the process model M_2 contains M''' which was not present in the process model M_1. Hence, there is a need to compute conformance score for such fragments of the process model. Moreover, even though the fragment M'', is present in both M_1 and M_2, the conformance score for such fragments needs to be recomputed, as the behavior between the activities present in such fragments may have changed after interactively modeling M_2 from M_1. However, the fragment M' is present in both M_1 and M_2, and the behavior between activities of this fragment is unchanged. Hence, the conformance scores calculated for M' in M_1 remains valid for M_2 too. Therefore, by keeping track of the changes in the process model, we can compute the projected conformance scores incrementally, and thereby re-use some of the prior conformance scores.

In the next section, we discuss the way in which the proposed approach has been realized. As discussed in Chapter 3 to 5, we use the class of labeled free-choice

workflow nets in combination with the synthesis rules in order to enable interactive process modeling. Hence, our solution for computing conformance is naturally tailored towards using the characteristics of free-choice workflow nets and the synthesis rules.

6.2 Projecting Labeled Free-choice Workflow Nets and Event Logs

In this section, we discuss the first aspect of our approach for computing conformance scores. We begin by discussing how we extract patterns from the event log, followed by extracting patterns from a labeled free-choice workflow net. Finally, we discuss a technique to compare the patterns extracted from the event log and the labeled free-choice workflow net in order to compute quantifiable conformance scores.

6.2.1 Extracting Patterns from Event Logs

The information from the event logs can be abstracted in multiple ways. In our approach, we calculate the so-called footprint patterns from the event log. The footprint patterns of an event log are composed of two types: (i) unary footprint patterns and (ii) binary footprint patterns.

Unary footprint patterns for an event log are obtained by projecting an event log on a single activity. A unary footprint pattern is a bag of projected traces on an activity.

Definition 42 (Unary footprint patterns in an event log)**.** *Let L be an event log and let $x \in \mathscr{A}$. Then the unary footprint pattern of activity x in L is the projection of activity x in L, i.e., $L\downarrow_{\{x\}}$.*

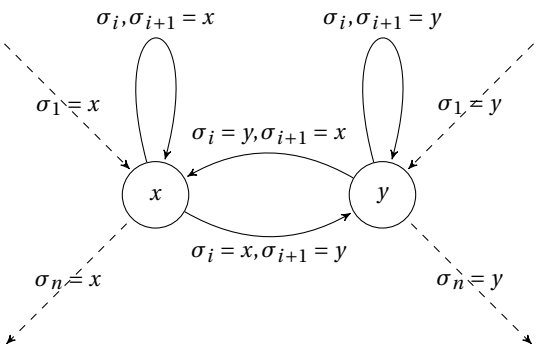
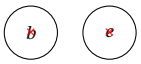
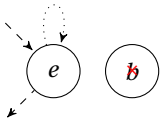
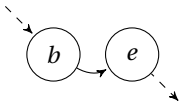
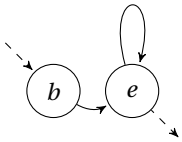


Figure 6.4: Binary footprint calculation mechanism for a trace σ.

Trace	Freq
$\langle a, a, f, c\rangle$	8
$\langle a, f, a, e, c\rangle$	1
$\langle a, f, a, e, e, c\rangle$	1
$\langle b, d, e, c\rangle$	4
$\langle b, d, e, e, c\rangle$	1
$\langle b, e, d, e, c\rangle$	4
$\langle a, a, f, e, e, e, c\rangle$	1

(a) An example of an event log.

Binary footprint pattern	Frequency
	8
	3
	4
	5

(b) Bag of footprint patterns for the complete log for the pair (b, e) from Figure 6.5a. The pair (b, e) is unordered.

Figure 6.5: Example event log and binary footprint patterns corresponding to (b,e).

We can project an event log on an activity by removing all the other activities from the event log. For example, the bag of unary footprint patterns of an activity e in the event log of Figure 6.5a is $[\langle\rangle^8, \langle e\rangle^5, \langle e, e\rangle^6, \langle e, e, e\rangle]$. The unary footprint pattern thus considers how often an activity is repeated a particular number of times.

Binary footprint patterns for an event log are used to calculate the relations between activities in a (unordered) pair-wise manner, for a given trace in the event log. We first discuss computation of binary footprint patterns for a trace.

Definition 43 (Binary footprint patterns in a trace)**.** *Let L be an event log and let $x, y \in \mathscr{A}$. Let $L' = L\downarrow_{\{\{x,y\}\}}$. Then the binary footprint pattern for the unordered pair (x, y) in a trace $\sigma \in L'$ is a graph $(\{x, y\}, S, E, F)$, where*

- *x and y are the nodes of the graph*
- $S = \sigma_1 \mid \sigma \neq \langle\rangle$
- $E = \sigma_{|\sigma|} \mid \sigma \neq \langle\rangle$
- $F = (a, b) \mid \exists_{1\leq i\leq|\sigma|}\sigma_i = a \wedge \sigma_{i+1} = b$

We introduce this using an example. Consider an *unordered pair* of activities (x, y) and a trace $\sigma = \langle\sigma_1, \sigma_2, ..., \sigma_n\rangle$, such that $\forall_{1\leq i\leq n}(\sigma_i = x \vee \sigma_i = y)$. Then, for any i, the binary footprint pattern of σ is calculated as shown in Figure 6.4.

An arc from the binary footprint pattern is removed, if the condition mentioned on the arc is not satisfied. The first activity of the trace σ can either be x or

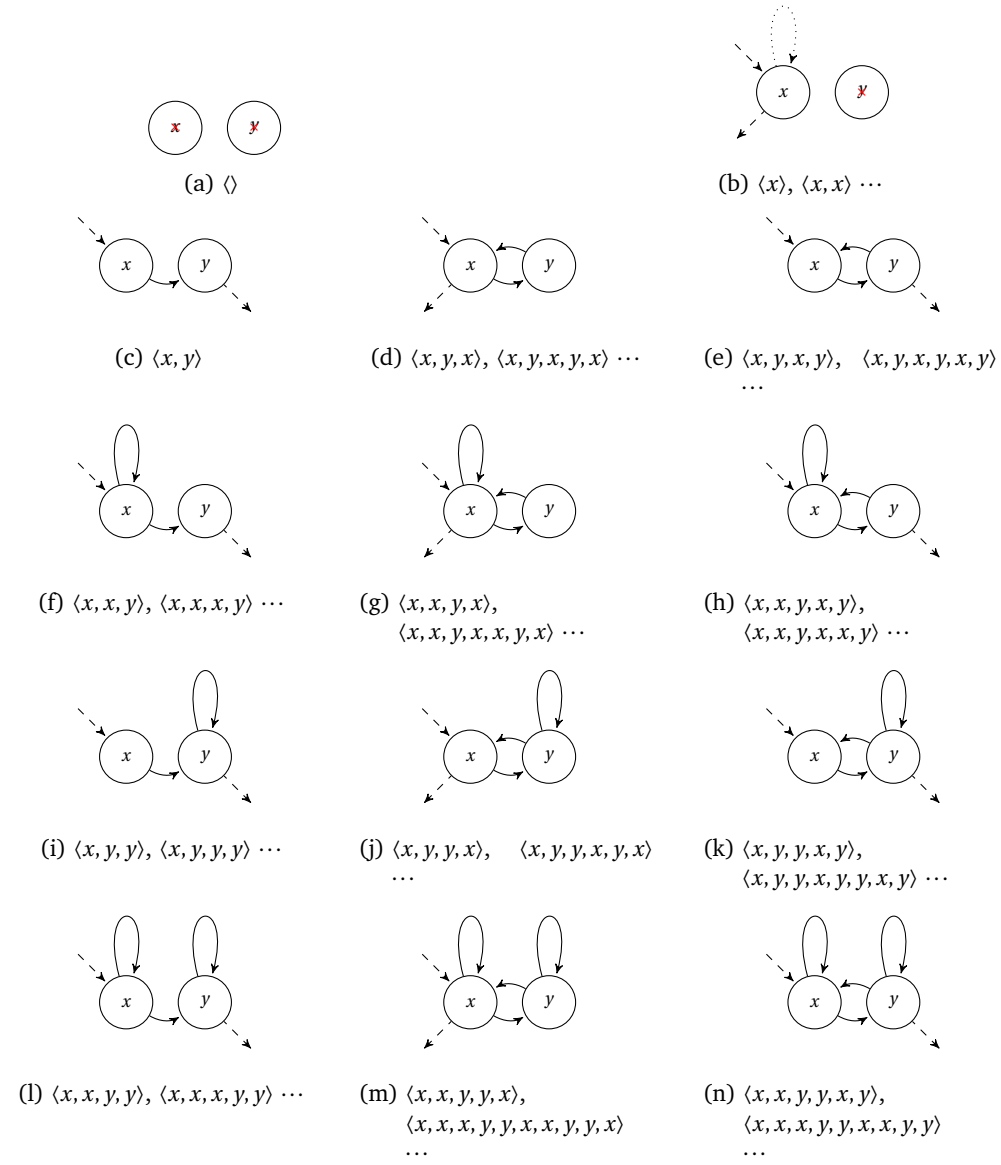


Figure 6.6: Binary footprint patterns when the traces start with x, except for the empty trace (6.6a) or when the trace contains only y's and no x's. For the latter, the pattern would be y-equivalent of 6.6b

y. Similarly, the last activity can either be x or y. Hence, for any given trace, there could only be two dashed arcs, one (incoming arc) related to the first activity, and one (outgoing arc) related to the last activity of the trace. The only exception to this is an empty trace, in which case no such arcs exists.

Figure 6.6 shows all the possible footprint patterns with some example traces, when the first activity of the trace is fixed to x, i.e., $\sigma_1 = x$, except for the empty trace (Figure 6.6a), or a trace which contains only y's. Note that for a given activity sequence we use the footprint pattern with the smallest number of arcs. For example, consider a trace $\langle x, x, y \rangle$, even though the footprint patterns shown in Figure 6.6l or Figure 6.6n are valid for such a trace sequence, Figure 6.6f is chosen as the appropriate footprint pattern as it contains the lowest number of arcs.

Unlike the unary footprint patterns, which record the number of times an activity is repeated, the binary footprint patterns do not record the number of times an activity is repeated. Also, when only one activity occurs in a trace sequence (Figure 6.6b), then the repetition of the activity is considered irrelevant for a binary footprint pattern. This is partly because, the repetitions of a single activity are already accounted for in the unary footprint patterns. We can thus obtain binary footprint patterns for all the unordered pairs of activities in the event log, by *projecting* the event log on pairs of activities.

In order to project an event log on a pair of activities, we simply remove all those activities from the event log which are not a part of the pair. Using such a projected event log, we can then compute the binary footprint patterns for all the activity pairs of the entire event log. Binary footprint patterns for the entire event log are constructed as a bag of binary footprint patterns. For example, consider a pair (b, e) from the event log of Figure 6.5a. The projected event log for the pair (b, e) is $[\langle\rangle^8, \langle e \rangle, \langle e, e \rangle, \langle e, e, e \rangle, \langle b, e \rangle^4, \langle b, e, e \rangle^5]$. The bag of binary footprint patterns of the pair (b, e) over the projected event log is shown in Figure 6.5b.

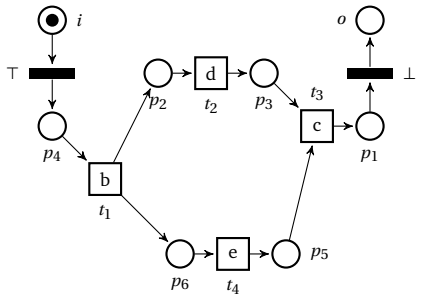
The footprint patterns of the event log are calculated only once. Now, we discuss how the footprint patterns are calculated in a labeled free-choice workflow net.

6.2.2 Extraction of Patterns from Labeled Free-choice Workflow Nets

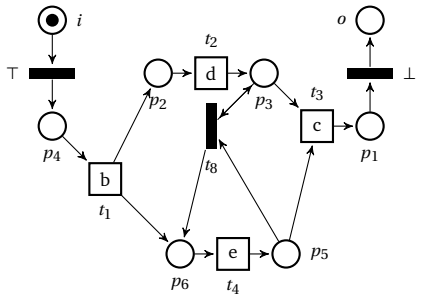
We now discuss the computation of unary and binary footprint patterns in the context of labeled free-choice workflow nets.

Unary footprint patterns for models In a labeled free-choice workflow net, the unary footprint pattern corresponding to an activity x, is a pair of values (M_x^{min}, M_x^{max}). M_x^{min} indicates the minimum number of times an activity x can occur in the process model, whereas M_x^{max} indicates the maximum number of times x can occur in the process model. M_x^{min} and M_x^{max} are calculated as follows:

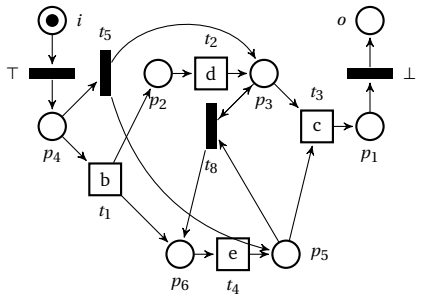
1. Project a labeled free-choice workflow net on x. This is done by making all the activities other than x invisible.
2. Consider a trace $\sigma = \langle x, \cdots, x \rangle$, where $|\sigma| = m$. Then we denote such a trace by σ^m.
3. Check whether it is possible to replay, i.e., reaching the accepting state of the labeled free-choice workflow net, using an empty trace $\sigma^0 = \langle\rangle$ on the net. If it is possible, then set $M_x^{min} = 0$. Otherwise, check if the replay of $\sigma^1 = \langle x \rangle$ is possible on the projected net. If yes, then set $M_x^{min} = 1$, and so on.
4. Next, count the number of visible transitions labeled with the activity x. Let this number be n. Check if any of the traces in $\{\sigma^{n+1} \cdots \sigma^{n+n}\}$ can be replayed on the projected net. If it is possible, then set $M_x^{max} = \infty$. This is because, at least one of the transitions labeled x appears twice, (as the net is free-choice), and hence it must be in a loop. Otherwise, check if



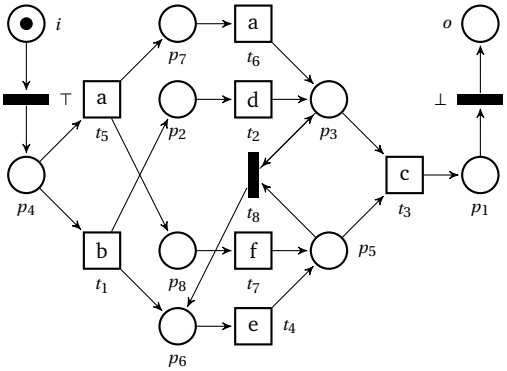
(a) Example labeled free-choice workflow net.



(b) Making e loop-able using ψ_T^{WF} to add a silent transition.



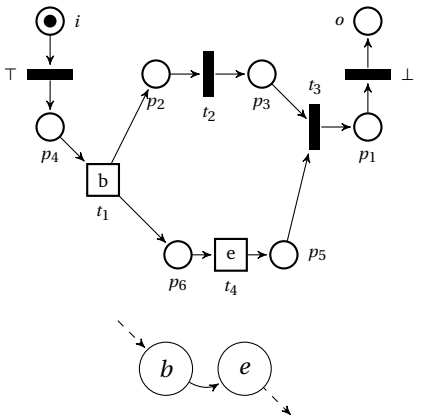
(c) Adding choice transition t_5 using ψ_T^{WF}.



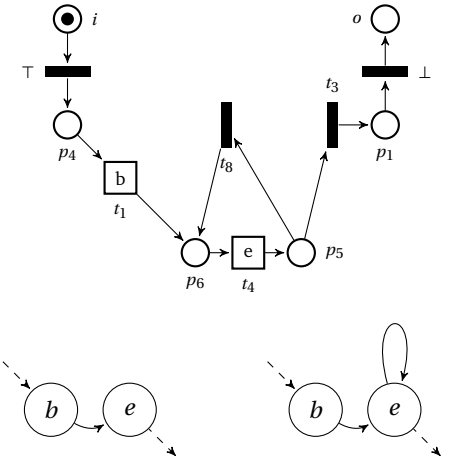
(d) Re-labeling t_5 as a and adding a duplicate a and f using ψ_A^{WF} twice.

Figure 6.7: Example synthesis of labeled free-choice workflow net corresponding to the event log from Figure 6.5a.

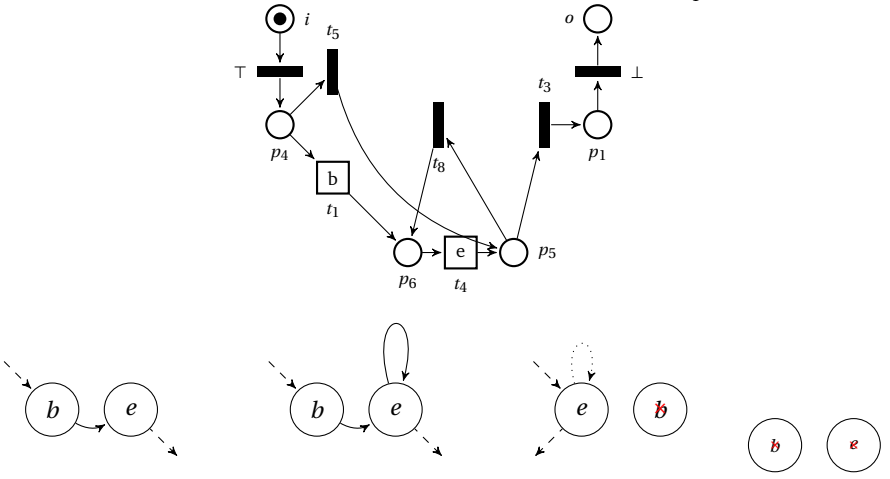
the replay of σ^n is possible on the projected net. If it is possible, then set



(a) Net projection and binary footprint pair corresponding to Figure 6.7a.



(b) Net projection and binary footprint pair corresponding to Figure 6.7b. Transition t_2 and place p_2 are removed using ψ_A^{WF} in reverse. Furthermore, place p_3 is removed by using ψ_P^{WF} in reverse.



(c) Net projection and binary footprint pair corresponding to Figure 6.7c (and also Figure 6.7d). In Figure 6.7c, transition t_2 and place p_2 is removed using ψ_A^{WF} in reverse. Place p_3 is removed by using ψ_P^{WF} in reverse. In Figure 6.7d, transitions t_6, t_7 and places p_7 and p_8 were additionally removed using the reverse of ψ_A^{WF} twice.

Figure 6.8: Set of binary footprint patterns corresponding to the activity pair (b,e). Note that the order of activities in the pair does not matter.

$M_x^{max} = n$, and so-on for $n-1, n-2, \cdots$.

The unary footprint pair for the labeled free-choice workflow net of Figure 6.7a corresponding to activity e is $(1,1)$ and the unary footprint pair corresponding to activity b is $(1,1)$ too. Whereas the unary footprint pairs for the labeled free-choice workflow net of Figure 6.7c corresponding to activity e is $(0,\infty)$ and the unary footprint pair corresponding to the activity b is $(0,1)$.

Binary footprint patterns for models are computed based on pairs of activity combinations, for all the activities present in the labeled free-choice workflow net. For any pair of activities, we calculate a set of binary footprint patterns *allowed* by a labeled free-choice workflow net. In order to calculate this set, we use the following steps:

1. Project the pair of selected activities on the labeled free-choice workflow net. This is done by making all the transitions which are not part of the activity pair invisible. For example, Figure 6.8a shows the projection for activity pair (b,e), on the labeled free-choice workflow net of Figure 6.7a. We can further use language preserving reduction rules, to reduce a labeled free-choice workflow net in order to remove unnecessary nodes [58].
2. Again, we use some minimal traces corresponding to each binary footprint pattern, and try to align them on the projected labeled free-choice workflow net. For example, in order to verify if the binary footprint pattern of Figure 6.6b holds for a labeled free-choice workflow net, a trace $\langle x \rangle$ is run over the labeled free-choice workflow net. If the process completes successfully, i.e., the end state is reached, then the binary footprint pattern as shown in Figure 6.6b is added to the set of binary footprint patterns for a pair. These minimal traces for each binary footprint pattern are the first traces of the corresponding binary footprint pattern from Figure 6.6.

The sets of binary footprint patterns for the activity pair (b,e) and the labeled free-choice workflow nets from Figure 6.7a, Figure 6.7b, Figure 6.7c and Figure 6.7d are shown in Figure 6.8. It should be noted that, we use a simple token-based replay approach for calculating the set of binary patterns that could be applicable in the net. This may not always yield a complete and/or correct set of patterns for all the free-choice workflow nets, which would require state-space exploration. However, we argue that in most circumstances the simple token-based approach suffices, especially when we consider the importance of time aspect in an interactive setting.

6.2.3 Comparing Labeled Free-choice Workflow Nets and Event Logs

We compare the footprint patterns of event logs and labeled free-choice workflow nets in order to calculate the conformance of an event log with a labeled free-choice workflow net, based on two metrics: *fitness* and *precision*. It should be noted that, in

Table 6.1: Fitness and precision scores for activity e based on the unary footprint patterns of the event log from Figure 6.5a and labeled free-choice workflow nets from Figure 6.7.

Net	L_e^1	M_e^1	f_e^1	$p^1_{max(e)}$	$p^1_{min(e)}$	p_e^1
Figure 6.7a	$[\langle\rangle^8, \langle e\rangle^5, \langle e,e\rangle^6, \langle e,e,e\rangle]$	$(1,1)$	$1/4$	$3/3$	$1/2$	0.75
Figure 6.7b	$[\langle\rangle^8, \langle e\rangle^5, \langle e,e\rangle^6, \langle e,e,e\rangle]$	$(1,\infty)$	$3/5$	$3/4$	$1/2$	0.625
Figure 6.7c	$[\langle\rangle^8, \langle e\rangle^5, \langle e,e\rangle^6, \langle e,e,e\rangle]$	$(0,\infty)$	1	$3/4$	1	0.875
Figure 6.7d	$[\langle\rangle^8, \langle e\rangle^5, \langle e,e\rangle^6, \langle e,e,e\rangle]$	$(0,\infty)$	1	$3/4$	1	0.875

the case of event logs, the footprint patterns are computed in terms of bags, whereas in the case of labeled free-choice workflow nets, the footprint patterns are computed in terms of sets (for binary footprints) or pairs (for unary footprints, i.e., a minimum and a maximum value for each activity).

Fitness

The fitness score reflects how much behavior from the event log is represented by the labeled free-choice workflow net. In our case, we first calculate the individual fitness of each activity based on the unary footprint patterns, as well as the fitness of activity pairs using the binary footprint patterns for all the activity combinations, and then aggregate the results to calculate the overall global fitness. It should be noted that, the fitness is computed only with respect to the activities which are already added in the labeled free-choice workflow net.

Fitness based on unary footprint patterns Let x be an activity from an event log which is added to the labeled free-choice workflow net. Let L_x^1 be the bag of unary footprint patterns from the event log for the activity x. Let $M_x^1 = (M_x^{min}, M_x^{max})$ be the unary footprint pattern from the labeled free-choice workflow net for the activity x. Then the fitness f_x^1 of the activity x based on unary footprint patterns is calculated as follows:

$$f_x^1 = \frac{\Sigma_{\sigma \in L_x^1 \wedge M_x^{min} \leq |\sigma| \leq M_x^{max}} L_x^1(\sigma)}{\Sigma_{\sigma \in L_x^1} L_x^1(\sigma)}$$

Using this, the fitness scores based on unary footprint patterns corresponding to activity e from Figure 6.5a and the labeled free-choice workflow nets of Figure 6.7 are computed as shown in Table 6.1.

Fitness based on binary footprint patterns Let (x, y) be an activity pair in the event log which is also present in the labeled free-choice workflow net. Let $L^2_{(x,y)}$ be the bag of binary footprint patterns from the event log for the activity pair (x, y). Let $M^2_{(x,y)}$ be the set of binary footprint patterns from the labeled free-choice workflow net for the pair (x, y). Then the fitness $f^2_{(x,y)}$ of the activity pair (x, y)

Table 6.2: Averaged fitness scores based on the unary and binary footprint patterns of the event log from Figure 6.5a and the labeled free-choice workflow nets from Figure 6.7a, Figure 6.7b and Figure 6.7c.

Activity (pair)	Figure 6.7a	Figure 6.7b	Figure 6.7c
b	9/20	9/20	20/20
c	20/20	20/20	20/20
d	9/20	9/20	20/20
e	5/20	12/20	20/20
(b, c)	9/20	9/20	20/20
(b, d)	9/20	9/20	20/20
(b, e)	4/20	9/20	20/20
(c, d)	9/20	9/20	20/20
(c, e)	5/20	12/20	20/20
(d, e)	4/20	9/20	20/20
Average Fitness	0.415	0.535	1

based on binary footprint patterns is calculated as follows:

$$f^2_{(x,y)} = \frac{\Sigma_{\sigma \in L^2_{(x,y)} \wedge \sigma \in M^2_{(x,y)}} L^2_{(x,y)}(\sigma)}{\Sigma_{\sigma \in L^2_{(x,y)}} L^2_{(x,y)}(\sigma)}$$

Using this, the fitness score $f^2_{(b,e)}$, for the activity pair (b, e) based on the binary footprint patterns of Figure 6.5b and labeled free-choice workflow nets from Figure 6.8a, Figure 6.8b, and Figure 6.8c are $1/5$, $9/20$ and 1 resp.

The fitness of the entire event log is calculated as the average of all the individual fitnesses for all the pairs of activity combinations (in the case of binary footprint patterns) as well as all the activities (in the case of unary footprint patterns). It should be noted that the fitness is computed only with respect to the activities that are present in the labeled free-choice workflow net. Table 6.2 shows the fitness scores for the nets from Figure 6.7a, Figure 6.7b and Figure 6.7c. As evident, when activity e cannot be repeated according to Figure 6.7a, the averaged fitness score is the lowest. When activity e can be repeated in Figure 6.7b, the fitness score is higher, however is not 1, as there exists no provision to *skip* activities b and d. However, in Figure 6.7c, when it is possible to skip both b and d, the averaged fitness is a perfect score of 1.

Precision

The precision score reflects how much extra behavior is allowed by a labeled free-choice workflow net, when compared with the event log. Similar to the fitness scores, we first compute the precision value of every activity and every activity pair by comparing the unary footprint patterns and the binary footprint patterns as follows:

Precision based on unary footprint patterns Let x be an activity from an event log. Let L_x^1 be the bag of unary footprint patterns from the event log corresponding to the activity x. Let $M_x^1 = (M_x^{min}, M_x^{max})$ be the unary footprint pattern from the labeled free-choice workflow net for the activity x. We compute precision based on two values, maximum occurrences of x allowed by the model, and minimum occurrences of x allowed by the model. $p^1_{max(x)}$ denotes the precision for maximum occurrences of activity x allowed by the model, and is calculated as follows:

$$p^1_{max(x)} = \frac{L_{max}}{M_m}$$

where, L_{max} is the length of longest trace in L_x^1, i.e., $L_{max} = |P|$ s.t. $P \in L_x^1 \wedge \forall_{P' \in L_x^1} |P'| \leq |P|$ and

$$M_m = \begin{cases} L_{max} & \text{if } L_m \geq M_x^{max} \\ L_{max} + 1 & \text{if } M_x^{max} = \infty \\ M_x^{max} & \text{otherwise} \end{cases}$$

In order to avoid division by ∞, we use an upper bound for M_m, as $L_{max} + 1$, when $M_x^{max} = \infty$. This also penalizes shorter loops observations from the event log, when compared to longer loop observations. However, there is also a drawback of the proposed solution. Consider two process models corresponding to the same event log. Let us assume that the value of $L_{max} = 5$ for a particular activity. Let us assume that the value of $M_x^{max} = 50$ for the first model and the value of $M_x^{max} = \infty$ for the second model. Based on the proposed solution, the value for the precision of the second model would be better than the value for the precision of the first model, which is incorrect. In our case, however, we are not interested in comparing the precision between process models. We are mainly interested in providing an approximate *indication* of the precision value. We re-visit this limitation in the evaluation section (Section 6.4.3).

The precision for minimum occurrences of x in the model is denoted by $p^1_{min(x)}$, and is calculated as follows:

$$p^1_{min(x)} = \begin{cases} 1 & \text{if } L_{min} \leq M_x^{min} \\ \frac{L_{min}+1}{M_x^{min}+1} & \text{otherwise} \end{cases}$$

where, L_{min} is the length of shortest trace in L_x^1, i.e., $L_{min} = |P|$ s.t. $P \in L_x^1 \wedge \forall_{P' \in L_x^1} |P'| \geq |P|$.

The overall unary precision for activity x is thus the average of the minimum and maximum precisions, denoted by p_x^1 and calculated as follows

$$p_x^1 = \frac{p^1_{max(x)} + p^1_{min(x)}}{2}$$

The precision scores corresponding to the activity e from the event log of Figure 6.5a and the labeled free-choice workflow nets from Figure 6.7 are shown in Table 6.1.

Table 6.3: Averaged precision scores based on the unary and the binary footprint patterns of the event log from Figure 6.5a and the labeled free-choice workflow nets from Figure 6.7a, Figure 6.7b and Figure 6.7c.

Activity (pair)	Figure 6.7a	Figure 6.7b	Figure 6.7c
b	0.75	0.75	1
c	1	1	1
d	0.75	0.75	1
e	0.75	0.625	0.875
(b, c)	1	1	1
(b, d)	1	1	1
(b, e)	1	1	1
(c, d)	1	1	1
(c, e)	1	1	1
(d, e)	1	1	1
Average Precision	0.925	0.913	0.988

Precision based on binary footprint patterns Let (x, y) be an activity pair. Let $L^2_{(x,y)}$ be the bag of footprint patterns from the event log for the activity pair (x, y). Let $M^2_{(x,y)}$ be the set of footprint patterns from the labeled free-choice workflow net for the activity pair (x, y). Then the precision $p^2_{(x,y)}$ of the activity pair (x, y) is calculated as follows:

$$p^2_{(x,y)} = \frac{\sum_{P \in M^2_{(x,y)} \wedge P \in L^2_{(x,y)}} 1}{|M^2_{(x,y)}|}$$

Using this, the precision score $p^2_{(b,e)}$, for the activity pair (b, e) based on the binary footprint patterns of Figure 6.5b and labeled free-choice workflow nets from Figure 6.8a, Figure 6.8b, and Figure 6.8c are all equal to 1.

The precision of the overall labeled free-choice workflow net with a given event log is then computed as the average of all the individual precision scores (like with fitness). Table 6.3 shows the precision scores for the event log from Figure 6.5a and the labeled free-choice workflow nets from Figure 6.7a, Figure 6.7b and Figure 6.7c respectively.

It should be noted that in the binary patterns for labeled free-choice workflow nets, our approach does not consider duplicate occurrences of activities. That is, by considering minimal traces in order to extract binary footprint patterns, the approach ignores the possible duplication of activities in the labeled free-choice workflow net. However, to a certain extent, we address this by using unary patterns for single activities, which consider frequencies from the event log, as well as the minimum and maximum possible occurrences from the labeled free-choice workflow net.

6.3 Incremental Tracking of Changes in Labeled Free-Choice Workflow Nets

Section 6.2 provides a way to calculate fast approximate conformance analysis, given a labeled free-choice workflow net and an event log. However, in an interactive process discovery setting, a labeled free-choice workflow net is expanded incrementally. In the second part of our solution, we make use of this principle in order to incrementally calculate the conformance using the projected conformance of the prior labeled free-choice workflow net. This aspect deals only with the comparison of binary footprint patterns of conformance analysis. Instead of recalculating the projected conformance of all the activity pairs, we calculate the projected conformance of only those activity pairs which are necessary, and re-use the previously computed projected conformance for all the other activity pairs.

Before introducing incremental conformance, we introduce the concept of behavioral equivalence in two models. Two labeled free-choice workflow nets M_1 and M_2 are said to be behaviorally equivalent, represented as $M_1 \approx M_2$ if and only if all the behavior of M_1 is exhibited by M_2, and vice versa. Similarly, behavioral in-equivalence of two labeled free-choice workflow nets is denoted by $M_1 \not\approx M_2$. Suppose a labeled free-choice workflow net M_{i+1} is interactively derived from a labeled free-choice workflow net M_i. Let C_{i+1} and C_i correspond to all the activity pairs of M_{i+1} and M_i. Let $M_i \downarrow^{(x,y)}$ denote projection of activity pair (x,y) on the model M_i. Then, we can distinguish two cases:

1. Set of *same* activity pairs $C_S \subseteq C_{i+1}$ whose projected behavior is the same in the models M_i and M_{i+1}, that is $\forall_{c \in C_S} M_i \downarrow^c \approx M_{i+1} \downarrow^c$.

2. Set of *different* activity pairs $C_D \subseteq C_{i+1}$ whose projected behavior is different in the models M_i and M_{i+1}, that is $\forall_{c \in C_D} M_i \downarrow^c \not\approx M_{i+1} \downarrow^c$.

There is no need to calculate the conformance values for those activity pairs which exhibit the same projected behavior in M_i and M_{i+1} (C_S). However, there is a need to calculate the conformance values for the activity pairs C_D whose projected behavior is *not* the same. Therefore, in an interactive setting, we improve the projected conformance calculation times, by calculating conformance for only the activity pairs from the set C_D. It should be noted that, if a new activity is added to the model interactively, then new activity pairs could be introduced, which would all be a part of C_D. The amount of time needed for the calculation of conformance for activity pairs C_S is saved by using an incremental way of calculating the conformance values.

Based on the type of synthesis rule used for interactive labeled free-choice workflow net construction at each iteration, we identify the set of pairs of activities (C_S) whose projected behavior does not change. For all the other activity pairs (C_D), the conformance is recalculated. We now describe the incremental way of calculating the set of activity pairs C_S and C_D, in the context of interactive process discovery based on each of the synthesis rules.

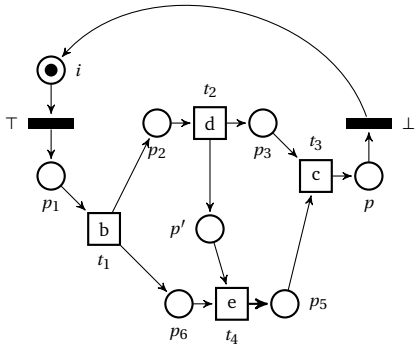


Figure 6.9: Adding a new place to the net from Figure 6.7a using ψ_P^{WF} rule. The result is that t_2 (d) and t_4 (b) are now in sequence (that is, d is followed by b).

6.3.1 Addition of a New Place (ψ_P^{WF})

Adding a new place using the ψ_P^{WF} rule to a net allows the possibility of introducing concurrency in the net. An introduction of a place does not result in any new activity in the model, and hence no new activity pairs are possible. There exists a set of bags of places P_{set} in the labeled free-choice workflow net, which has the same effect as the newly added place. This can be extracted from the linear dependence condition of the ψ_P^{WF} rule (and the λ vector). Loosely speaking, this means that every bag of places from P_{set} collectively has the same input and output as the newly added place. This set corresponding to the place p_5 is $\{[p_2, p_3]\}$ in the net of Figure 6.10a before using ψ_A^{WF} to add p_6 and t_4. Typically, all the activity combinations are added to C_S, as the projected behavior of the activity pairs remains unchanged. However, in very few cases the projected behavior of the activity pairs might have changed, if at least one of the bags in P_{set} contains the place i of the labeled free-choice workflow net. For example, in Figure 6.9, P_{set} corresponding to the newly added place p_7 is $\{[p_3, p, i, p_1, p_6]\}$. Since P_{set} contains i, no activity pairs are added to C_S. This is also because the projected behavior between some activities has indeed changed as shown in Figure 6.9 (for example, t_2 and t_4 changed from parallel to a sequential construct).

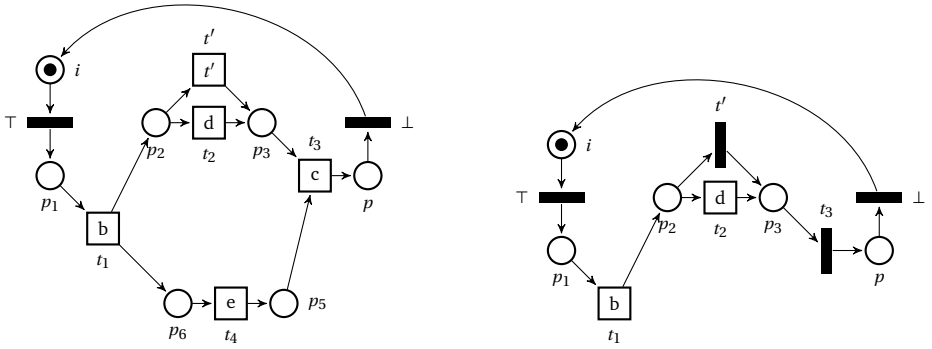
6.3.2 Addition of a New Transition (ψ_T^{WF})

The addition of a new transition using the ψ_T^{WF} rule usually results in the introduction of a choice or a loop in the labeled free-choice workflow net. There exists a set of bags of transitions T_{set}, which have the same effect on the labeled free-choice workflow net, as the newly added transition. For example, in Figure 6.7c, this set is $[\{t_1, t_2, t_4\}]$ corresponding to the linearly dependent transition t_5. Similarly, in Figure 6.10, this set corresponding to linearly dependent transition t' is $[\{t_2\}]$. We use this information to calculate the set of activity combinations C_S whose projected behavior does not

change. No elements are added to C_S, if any bag from T_{set} contains $\top$ or $\bot$. The reasons for this are similar to Section 6.3.1. The second scenario is when none of the bags from T_{set} contain $\top$ or $\bot$, i.e., $\forall_{E \in T_{set}} \top, \bot \notin E$. Let T_L be the set of all the labels represented by the transitions in T_{set}. An activity combination for a subset of activities A_s (such that the label of newly added transition is not in A_s) is added to C_S if $A_s \cap T_L = \emptyset$. Consider the labeled free-choice workflow net from Figure 6.10a derived from Figure 6.7a by adding a new transition t', which is also labeled t' ($T_L = \{t'\}$). The pair of activities (b, d), (c, d) and (e, d) are not added to C_S. For example, consider the projection of activities $\{b, d\}$ as shown in Figure 6.10b. In the new net projected (and reduced) on activities (b, d), there is a possibility to *skip* the activity d, via t'. Hence there is additional behavior introduced corresponding to the activity d which was not present in the prior projected net. Hence such activity pairs are not added to C_S, and are candidates for recalculation. As a counter-example, it is easy to see that the previous activity pair of (b, e) has the same projected behavior, after the introduction of t', and hence this activity combination is added to C_S.

6.3.3 Addition of a New Transition and a New Place (ψ_A^{WF})

Adding a new transition and a new place using the ψ_A^{WF} rule results in a new sequence in the model. If the newly added transition is labeled with an activity which is not already present in the labeled free-choice workflow net, then all the activity pairs from the previous net are behaviorally equivalent in the newly synthesized net. That is, if the newly added transition is made silent, then the net would be behaviorally equivalent to the previous net. Hence all the activity pairs from the prior net are added to C_S. In case the newly added transition is labeled with an activity which was already present in the prior net, then the conformance is computed only for the



(a) An example application of the ψ_T^{WF} rule to add t' to the net from Figure 6.7a.

(b) Net projected on activity pair (b,d)..

Figure 6.10: An example of the ψ_T^{WF} rule and the usage of incremental conformance calculation.

Table 6.4: Characteristics of the real-life event logs chosen for evaluation.

Event log	Number of cases	Number of events	Number of activity classes
Sepsis	1050	15214	16
BPI-2012A	13087	60849	10
BPI-2012O	5015	31244	7
BPI-2015	696	12559	25
Hospital billing	100000	451359	18
BPI-2018	15364	25142	10

activity pairs corresponding to the activity of the newly added transition.

6.4 Implementation and Evaluation

The proposed approach is implemented as a plug-in to extend the capabilities of interactive process modeling proposed in Chapter 5. The user is provided with the fitness and precision scores, which are indicated on a scale from 0 to 1. The plug-in is available in the nightly build version of ProM [171], in the *InteractiveProcessMiningLite 6.9.13* package, under the plug-in "Petri net editor with conformance". This plug-in requires an event log in XES [3] format as input along with another object called AHL (Aggregated Log Heuristics), which contains information about unary and binary patterns of the event log. The AHL object can be obtained by using the plug-in "Log to Aggregated Log Heuristics for fast conformance checking" which requires an event log in XES format as an input.

Another stand-alone version of the proposed conformance approach is also available in the plug-in "Log-Model comparison using patterns". This plug-in requires an event log in XES [3] format and a Petri net as input.

The stand-alone version of the proposed approach has been evaluated based on performance aspect and accuracy aspect, by comparing it with the relevant state-of-the-art approaches. The techniques used for comparing fitness scores are: the decomposed replay technique [180], the recomposed replay technique [178], the projected conformance checking [112] framework (with $k = 2$), and the alignment-based conformance technique [5]. The techniques used for comparing precision scores are: the projected conformance checking [112] framework (with $k = 2$), and the escaping edges based ETC 1-align precision [128]. We performed the evaluation using six publicly available real-life event logs: (i) the Sepsis event log[1] containing the workflow data for roughly 1000 patients suffering from Sepsis in a hospital, (ii) the BPIC 2012-A[2] event log and (iii) the BPIC 2012-O[3] event log which correspond to the application and the offer sub-logs respectively of a loan application process in a Dutch financial

[1] https://data.4tu.nl/repository/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460
[2] https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
[3] https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

institute, (iv) the BPIC 2015[4] filtered event log, which contains top 25 activities of a building permit application for one of the municipalities in the Netherlands, (v) the hospital billing[5] event log, containing data from the financial modules of the ERP system of a regional hospital, and (vi) the BPIC 2018[6] event log, containing 10% of the data about the process for direct payments for German farmers. We used the inductive miner-infrequent algorithm [110], in order to discover labeled free-choice workflow nets from these event logs. By default, discovery techniques such as the inductive miner and the ILP miner [170] aim at discovering fully fitting labeled free-choice workflow nets, i.e., nets which have a fitness value of 1. As we are interested in comparing the fitness and precision values across different conformance algorithms, we configured the inductive miner infrequent algorithm to be used in three settings: (i) *ind0:* a setting assuming 0 noise, which guarantees a fitness of 1, (ii) *ind5:* a setting with noise threshold set at 50% noise, and (iii) *ind10:* a setting with noise threshold set at 100%. It should be noted that inductive miner discovers constructs such as loops, choices, concurrency and silent activities. However, the inductive miner cannot discover duplicate activities in a labeled free-choice workflow net.

6.4.1 Performance

The primary objective of the approach proposed in this paper is to enable *faster* conformance analysis which would be useful in an interactive process discovery setting. The extraction of footprint patterns (unary and binary) from the event logs would serve as an input to the conformance checking during interactive process discovery, and is required to be done only once. Hence, we did not use this processing step during comparison. This initial step took in between $10ms$ to $2500ms$, depending on the type of event log. Contrary to this, other conformance techniques require the usage of the complete event log during each step of interactive process discovery. Figure 6.11 shows the performance of different approaches, in milliseconds. Note that the scale is logarithmic. The computation time for the PCC framework [112] and our approach includes the calculation time for both the fitness and precision scores. The decomposed replay, recomposed replay and alignment-based replay are used to calculate only the fitness value. The ETC 1-align technique is used to calculate the precision value. It should be noted that, the ETC 1-align technique requires an alignment as an input for calculating the precision value. However, the plots of Figure 6.11 do not show the time taken for computing alignments in the case of ETC 1-align. It is easy to see that our approach is able to outperform most of the techniques for calculating faster conformance, for both fitness and precision scores. The PCC framework is the closest in terms of computation time compared to our approach, and it can be argued that both our approach and the PCC framework scale similarly. Waiting times of most state-of-the-art techniques can be greater than 10 seconds, and hence become unacceptable. Compared to this, our approach works much faster, and hence is suitable in an interactive process discovery setting.

[4] https://doi.org/10.4121/uuid:a0addfda-2044-4541-a450-fdcc9fe16d17

[5] https://doi.org/10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741

[6] https://data.4tu.nl/repository/uuid:3301445f-95e8-4ff0-98a4-901f1f204972

6.4.2 Fitness and Precision Comparison

In this sub-section, we compare the outcome of our result, in terms of fitness and precision scores, by comparing it with other approaches. All the techniques considered in this evaluation, calculate the fitness and precision scores in the (inclusive) range of 0-to-1. Higher scores indicate better fitness/precision. Hence, a fitness/precision score of 1 would indicate a perfect fitness/precision. Moreover, the decomposed and recomposed replay techniques do not give a singular value for fitness scores, but instead give a lower bound and higher bound for fitness. The comparison outcome of fitness and precision scores of all the techniques is shown in Figure 6.12 and Figure 6.13 resp. As discussed earlier, we considered three labeled free-choice workflow nets for each type of event log discovered using the inductive miner infrequent at various settings. The inductive miner infrequent guarantees that a labeled free-choice workflow net discovered with a setting of no noise (i.e., noise threshold set to 0), are perfectly fitting with the event log. This is also evident in all the fitness scores of all the labeled free-choice workflow nets. As the noise threshold is increased, the inductive miner allows models that explain just the common part of the log. Hence while increasing the noise threshold the fitness should drop. This is indeed the trend that is observed across the fitness measures for all the techniques (including ours). Moreover, it can be seen that for most of the scenarios, the fitness scores of our technique are extremely

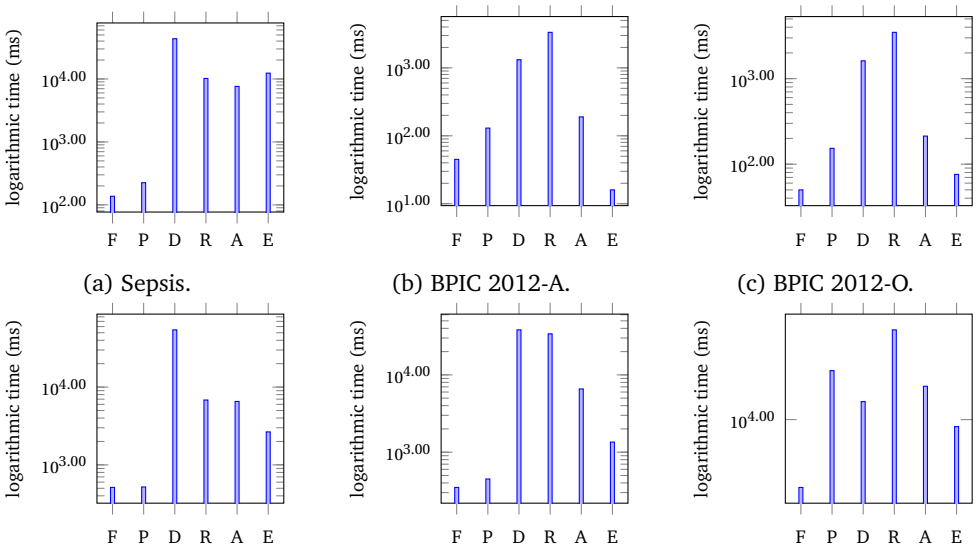
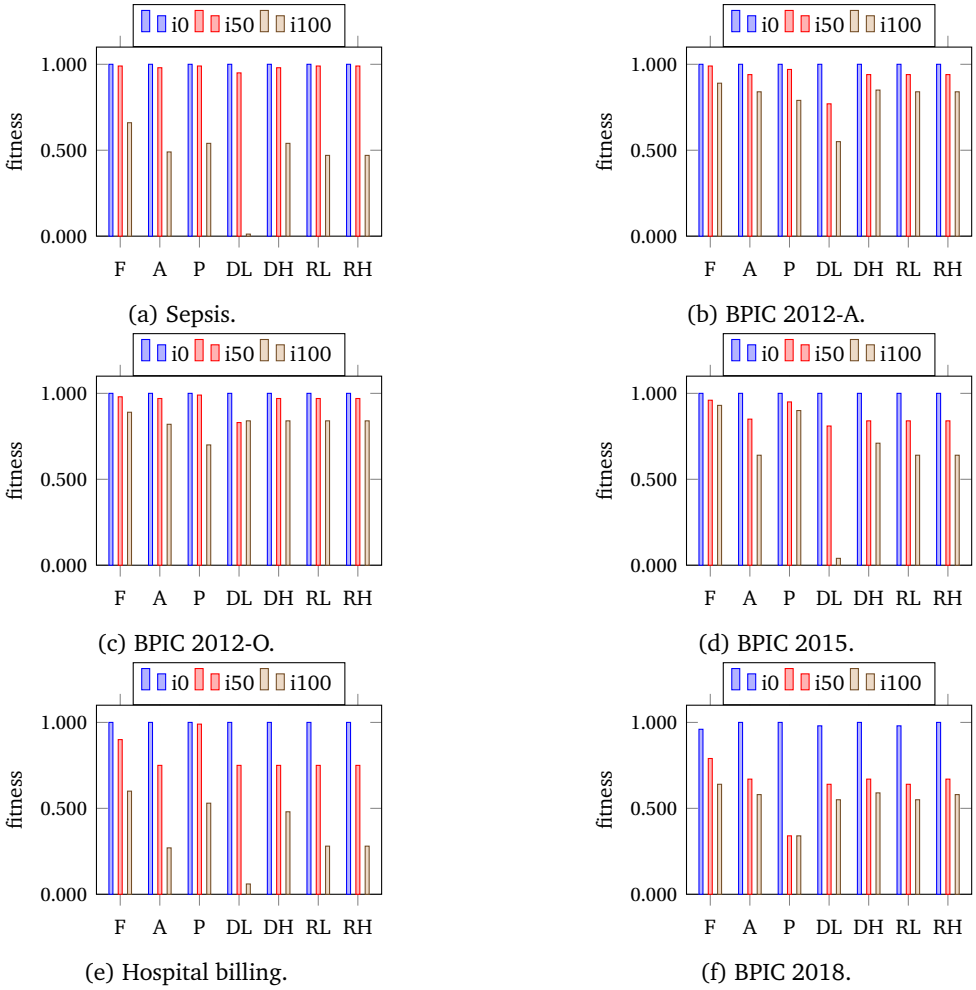


Figure 6.11: Time (logarithmic) comparison for various approaches: F - Footprint-based Conformance Technique - proposed in this chapter, P - PCC framework ($k = 2$), D - Decomposed Replay, R - Recomposed Replay, A - Alignment-based replay, E - ETC 1-align precision.



(a) Sepsis.

(b) BPIC 2012-A.

(c) BPIC 2012-O.

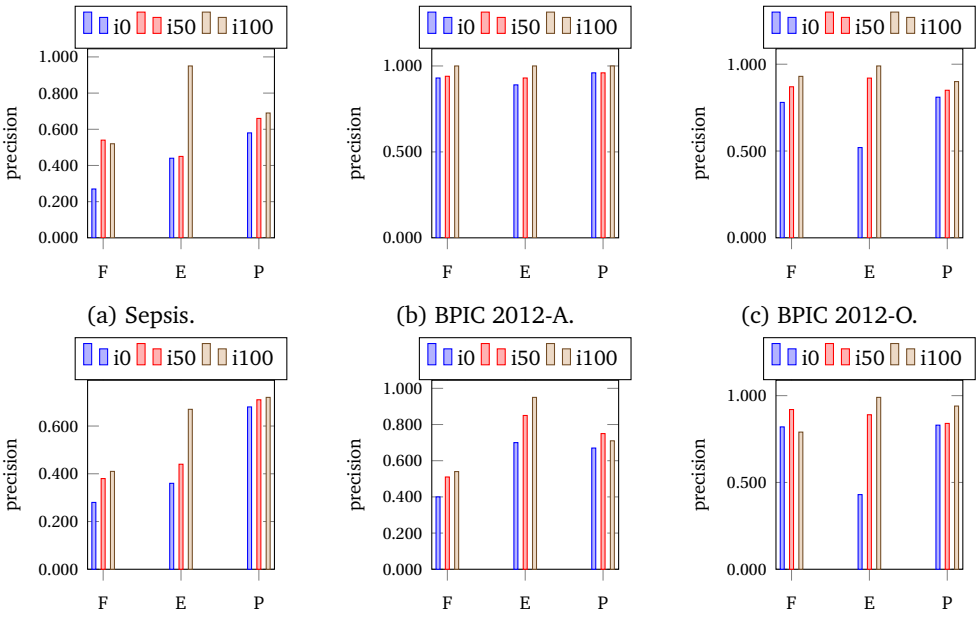
(d) BPIC 2015.

(e) Hospital billing.

(f) BPIC 2018.

Figure 6.12: Fitness values comparison of: F - Footprint-based Conformance Technique - proposed in this chapter, A - Alignment-based replay, P - PCC framework ($k = 2$), DL - Decomposed Replay Lower Bound, DH - Decomposed Replay Higher Bound, RL - Recomposed Replay Lower Bound, RH - Recomposed Replay Higher Bound. The legends indicate the labeled free-choice workflow nets considered for calculating fitness, discovered using the inductive miner infrequent technique, such that: (i) *i0* - noise threshold set to 0, (ii) *i50* - noise threshold set to 50% (iii) *i100* - noise threshold set to 100%.

close to the alignment-based replay technique, which can be considered as a baseline for fitness scores. In a similar way, it is quite intuitive to note that as the labeled free-choice workflow nets become more restricted (with higher noise threshold), the precision of the labeled free-choice workflow net increases. The PCC framework and



(a) Sepsis. (b) BPIC 2012-A. (c) BPIC 2012-O.

(d) BPIC 2015. (e) Hospital billing. (f) BPIC 2018.

Figure 6.13: Precision values comparison of: F - Footprint-based Conformance Technique - proposed in this chapter, P - PCC framework ($k = 2$), E - ETC 1-align. The legends indicate the labeled free-choice workflow nets considered for calculating precision, discovered using the inductive miner infrequent technique, such that: (i) *i0* - noise threshold set to 0, (ii) *i50* - noise threshold set to 50% (iii) *i100* - noise threshold set to 100%.

our approach typically have a similar precision value, as shown in Figure 6.13, This is due to the similarity of the two approaches in terms of calculating the precision scores. It can be argued that there is no single baseline available when it comes to precision scores. However, it can be observed that all three approaches follow a similar trend in precision scores. Hence, in a practical scenario, our approach is useful as it provides an approximated, yet acceptable, indication of the precision and the fitness scores.

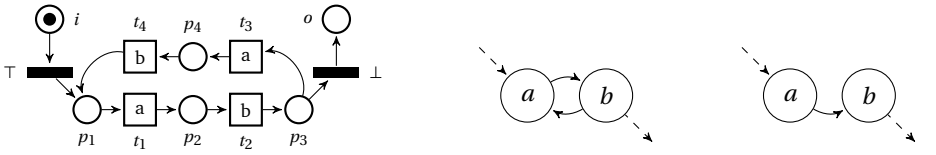
6.4.3 Discussion

The solution proposed in this chapter is suitable for the computation of the conformance scores in an interactive setting. However, the proposed solution has some limitations, which could be improved in the future. In this section, we discuss some of the possible issues with the proposed solution. Note that it is not the intention here to compare the proposed solution with the works from the literature, which are addressed in Section 6.5.

First, our solution faces difficulties when dealing with duplicate occurrences of activities in a process model. In particular, we do not consider duplicate occurrences when using the binary footprint patterns for computing fitness or precision. For example, consider the process model from Figure 6.14a, containing duplicate occurrences of activities a and b. Clearly the correct binary pattern for this process model is shown in Figure 6.14b. However, the binary pattern extracted using a minimal trace-based approach ($\langle a, b \rangle$) would be the one shown in Figure 6.14c. Since the extracted pattern is incorrect, this may affect the overall fitness and precision scores. A way to address the issues created by using the minimal trace-based approach, would be to compute the complete state-space of the projected model. However, exploring the complete state-space would eventually have a negative effect on the performance of the system. Another consequence of not distinguishing duplicate occurrences would be that multiple occurrences of activities within the process model would have identical fitness and precision values. We argue that, to a certain extent, the duplicate occurrences are addressed while computing the fitness and precision values using the unary footprint patterns.

Another possible issue in our approach is the computation of possibly incorrect precision scores as discussed in Section 6.2.3. This could also be the reason for relatively higher differences in the precision scores computed using our solution and the state-of-the-art, as shown in Figure 6.13. In the process mining community, the computation of precision scores has been an issue of much interest of late. However, in our case, we are only interested in providing an indication of the precision, and do not claim any guarantees with respect to the computed precision value. More importantly, we focus on the performance aspect, to compute the conformance scores in a speedy way.

Overall, even though the results were approximated in the proposed solution, the fitness and precision scores computed using our solution still provided a good indication in comparison to the fitness and precision scores as computed by other approaches. Moreover, by abstracting the event log using the footprint patterns, our solution was able to compute these results in a much faster way, and hence is better suited in an interactive setting. Nonetheless, it would be ideal to explore further avenues using the proposed approach, to improve some of the limitations in the proposed solution.



(a) An example net with duplicate activities. (b) Correct pattern. (c) Pattern extracted using minimal trace.

Figure 6.14: Extracting binary footprint patterns from the process model containing duplicate activities.

6.5 Related Work

The work presented in this chapter centers around proposing faster conformance analysis in order to enable conformance analysis in an interactive process discovery setting. Since the scope is conformance analysis rather than process discovery itself, in this section we discuss techniques from literature which propose conformance analysis in the context of process mining, that are comparable to our technique. Conformance checking techniques typically match the behavior of event logs with the behavior as depicted by process models. For a general overview on conformance analysis in process mining, we refer to [34].

6.5.1 Token-based Replay

[55, 142] proposed conformance checking in the context of process mining using the token-based replay in Petri nets. Every trace from the event log is replayed on the Petri net, such that whenever there are insufficient tokens to execute an event, it is recorded as a *missing token* and the event is executed anyway. When the trace terminates, the number of *missing tokens* and *remaining tokens* from the net are used to compute fitness and precision scores. The originally proposed token-based replay techniques had difficulties in handling silent and duplicate activities. A couple of closely related approaches, [174, 182] overcome such drawbacks and improve the fitness, precision and generalization scores by introducing negative events in the event log. These approaches could however have long running times. In all these approaches, the complete event log needs to be compared with the process model in order to compute the conformance score. Even though token-based replay technique are fast, for very large event logs, the compute times could still be high. In our proposed solution, we use a minimal trace based replay approach, to compute the binary footprint patterns of the process model, which is similar to the token-based replay approach. However, we stop the execution if the next event cannot be replayed, and do not have the notion of missing and remaining tokens. If the token-based replay technique could work with some abstracted version of the event logs, then it could possibly be used with the generic approach proposed in Section 6.1.

6.5.2 Alignment-based Conformance

Alignment-based conformance checking techniques such as [4,5,31,38,50] have been proposed that aim to optimally align the behavior of traces from the event log and the possible process model execution traces. Multiple techniques have been proposed to improve the proposed alignment approach, by translating the alignment problem into a planning problem [53], or assuming partially ordered input data [114], or constructing cost function for alignments based on attribute data [7]. However, the main focus of most of these techniques is to provide accurate conformance results. And hence, there is not a lot of emphasis on the performance of the technique itself. Furthermore, these techniques require scanning of the complete event log in order to perform conformance analysis. This is not ideal in a process discovery setting which

is interactive in nature and would require conformance checking after each step, as the performance of the overall system could be drastically impacted depending on the size of the event log and the complexity of the labeled free-choice workflow net being modeled. There have been specialized strategies proposed to incrementally repair alignments in the context of Evolutionary Tree Miner (ETM) algorithm [177]. However, the class of process models supported is limited to block-structured process models (process trees).

6.5.3 Divide-and-conquer

There are also techniques proposed in order to improve the performance of calculating conformance of a process model and an event log. Many of these techniques use the so-called divide-and-conquer strategy [63, 129, 155, 162, 178, 180]. The idea behind these techniques is to decompose a process model into various sub-models based on a specific decomposition strategy, and then to compute alignments on the smaller model (and a corresponding smaller event log) and aggregate the information across all the models. Another strategy to improve the computation time of conformance analysis is by using the Projected Conformance Checking (PCC) technique [112], which projects process models and event logs onto a subset of activities, and aggregates the results over all combinations of activity subsets. Unlike the PCC technique, we first abstract the information from the event logs, and then use this abstracted information in order to calculate conformance on a set of projected activity combinations. We argue that our approach is faster by not having to scan through the entire event log during every stage (step) of the interactive process discovery. This distinguishes our approach considerably from most of the other approaches too. Computing conformance scores using the abstractions on the process model and the event log have also been proposed in the literature. For example, [143] uses behavioral appropriates and footprint matrices to compare directly following and eventually following graphs. However, this could be infeasible in practice, as it requires exploration of the full state-space. [183, 184] propose computing the process compliance based on behavioral profiles, by comparing behavioral constraints in pairs of activities in the process model and the event logs, thereby overcoming the problems of exploring state-spaces in order to compute conformance. Hence, this approach is also closely related to our approach. Both our approach and the behavioral-profiles-based approach face difficulties in dealing with duplicate occurrences of activities in the process model. In essence, we argue that the behavioral constraints taken into account by the behavioral-profiles-based approach are a subset of the footprint patterns proposed by us. Another distinguishing factor of our approach is the calculation of conformance scores in an incremental fashion, by considering only those parts of the labeled free-choice workflow net which are changed, which can be derived based on the type of the synthesis rule used. However, the generic approach proposed in Section 6.1, could possibly be combined with any other approach from the literature which could work with abstracted event logs.

6.5.4 Other Techniques

Rule-based conformance checking techniques, such as [12, 35, 115, 136], verify if certain pre-specified rules were satisfied in reality, on a rule-by-rule basis. Our approach uses a similar principle. In essence, we abstract patterns, similar to rules, from both the event logs and the process models, and then incrementally compare them when a process model is interactively changed to compute the conformance scores. Some techniques address conformance checking from other perspectives such as natural language processing, formal methods, real time setting etc. [6, 154, 173]. Similarly, certain visualization driven approaches, allow the user to search the event log through visual interfaces by using approaches such as time-interval based queries and regular expressions [29, 107, 194]. The outcome of the search results could be used to analyze any compliance issues in the event logs. However, the goal of these techniques is different compared to our technique, as we are interested in computing conformance scores for procedural models (labeled free-choice workflow nets) in an efficient way.

6.6 Conclusion and Future Work

In this chapter, we presented an approach to enable fast conformance analysis by abstracting the information from the event logs using footprint patterns. Furthermore, we presented a way to extract similar footprint patterns from the process models, and also presented a way to compare the footprint patterns from the event logs with the footprint patterns from the process models, in order to deduce the fitness and precision scores. We also presented a technique to incrementally keep track of the

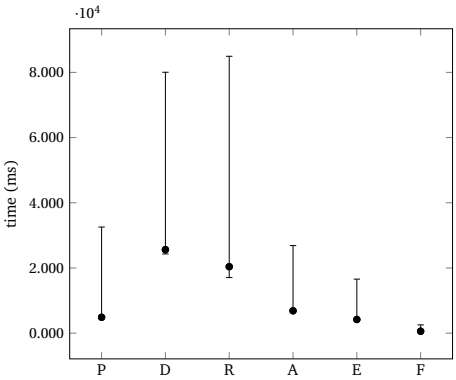


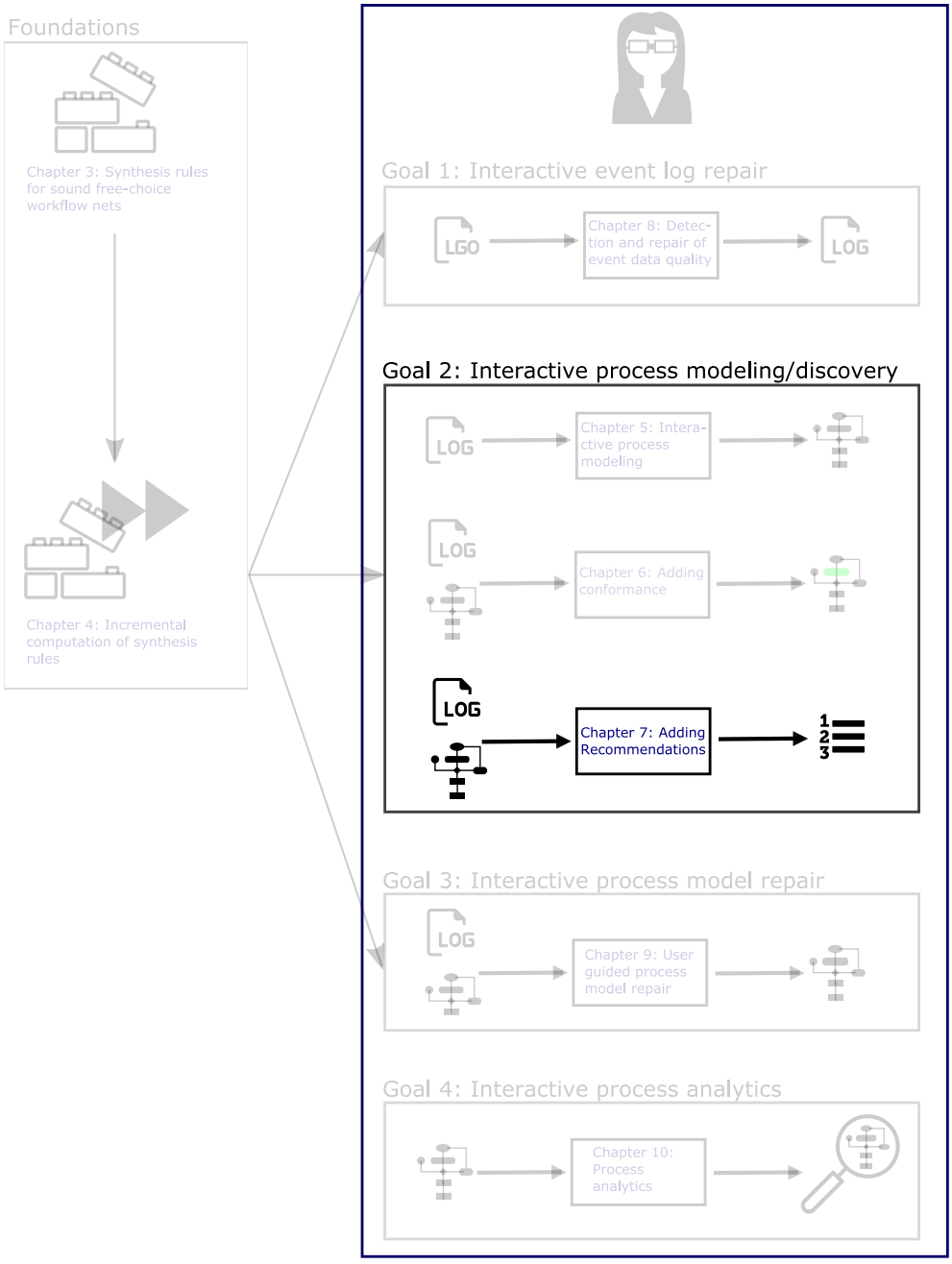
Figure 6.15: Average, minimum, and maximum time taken for computing conformance (fitness and/or precision) using various state-of-the-art techniques (P - PCC framework ($k = 2$), D - Decomposed Replay, R - Recomposed Replay, A - Alignment-based replay, E - ETC 1-align precision, F - Footprint-based Conformance Technique - proposed in this chapter.), based on the six real-life event logs and their corresponding discovered process models discussed in Section 6.4.

changes in the process model to further improve the efficiency of conformance analysis. As shown in Section 6.4, by reusing the precalculated footprint patterns of the event log in an incremental way, we were able to improve the performance times of calculating the conformance between a process model and the event log. Furthermore, the approximated conformance results calculated using our technique are comparable to many state-of-the-art techniques. Hence, we argue that our technique is much more suited for computing conformance in the context of process discovery techniques which are interactive in nature, wherein the event log remains unchanged in different steps/stages of the process discovery. The benefits of the proposed solution are clearly evident upon revising Figure 6.2, by including the solution proposed in this chapter as shown in Figure 6.15. Since the proposed solution can compute conformance scores in a much faster way (less than 500 milliseconds on an average), it is well suited in an interactive setting. However, the proposed solution also has some limitations, as shown in Section 6.4.3.

In the future, we would like to extend our approach in order to consider the frequencies of loops in binary footprint patterns directly. Furthermore, it would also be desirable to explore solutions that deal better with duplicate occurrences of activities in the process model. In the current solution, duplicate occurrences of activities and loops in the process model could result in the computation of incorrect fitness and precision scores. Currently, we trade the accuracy of our results in lieu of better performance. However, in the future it would be ideal to explore avenues that can improve the accuracy of the result, without impacting the performance of the system. It would also be desirable to give certain guarantees with respect to the fitness and precision scores. Another future direction could be, to allow the user to add or remove footprint patterns, in order to influence the conformance computation, by incorporating domain knowledge.

Chapter 7

Recommendations for Interactive Process Discovery



This chapter is based on the publication [59].

As discussed previously, automated process discovery and process modeling form two ends of a spectrum. On one end of the spectrum, automated process discovery techniques discover a process model directly from the event log with very limited user input. The user has very little influence over the actual discovery algorithm, and it is usually not possible to incorporate domain knowledge during process discovery. Moreover, the resulting process models might be incomprehensible for the end user. On the other end of the spectrum, we have the editor-based process modeling tools which are completely user-driven and use no historical evidence from the event logs to create the process model. We have minimized this gap in Chapter 5, by allowing the users to interactively construct process models, guided by the information from the event logs as projected on the process model. Moreover, in Chapter 6 we have further assisted the user in effective process modeling by performing real-time conformance analysis to indicate the usefulness of a change in the process model.

However, in the approaches proposed in Chapter 5 and Chapter 6, the user has to first determine the positioning of activities in a process model, based on the event log projections and/or domain knowledge, and then model the process model accor-

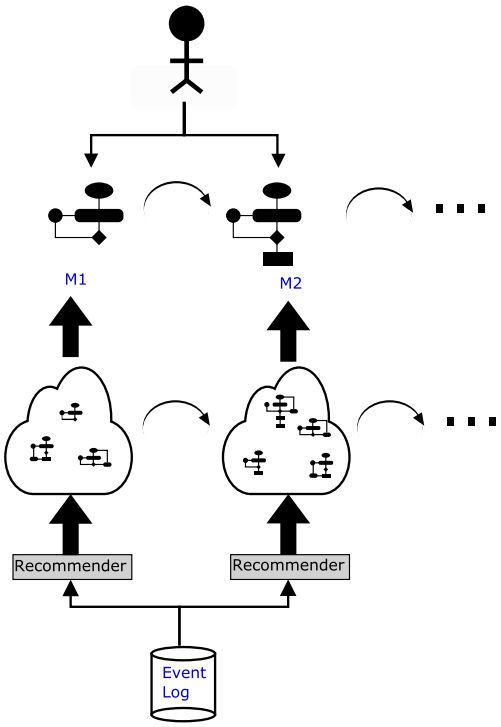


Figure 7.1: Recommendations-based interactive process discovery. Recommendations are generated to position an activity in a process model based on the event log. The user may then choose one of the suggested recommendations, or can ignore all the recommendations to manually position the activity in question.

dingly. In certain situations, the user may be unsure of where to position a particular activity in the process model. This is especially true when the user's knowledge about the activity in the process is limited. Moreover, there may be scenarios in which the information from the event log seems unclear for interpretation or decision making. Hence, it would be ideal to automatically discover process models on behalf of the user. This points back to the world of automated process discovery. However, the drawback of automated process discovery techniques is that they are typically black boxes, and allow for very limited user influence. That is, in an automated process discovery setting, the user is typically provided with one final process model. However, in our case, we would like the user to actively influence the discovery process, and be provided with multiple options to choose from. Hence, there is a need for providing smart recommendations to mitigate the burden of decision making from the user. In this chapter, we focus on addressing this issue by further reducing the gap between process modeling and automated process discovery. Figure 7.1 provides an overview of recommendations-based process modeling. The main contributions of this chapter are:

- A seamless integration of user-driven and data-driven approaches for process discovery, by providing recommendations to the user about the possible positions in a process model where an activity could be placed. Furthermore, it is also desirable to analyze the impact of adding the activity at the specified location in the process model.
- The possibility of an auto-complete: to switch efficiently between interactive discovery and automated discovery of a process model.

We begin by defining the problem in Section 7.1. We discuss the key aspects of the solution in Section 7.2, followed by the interface and interaction component of the system in Section 7.3. In Section 7.4 we evaluate the proposed approach to judge the quality of the recommendations based on a user study. Finally, we provide the conclusions in Section 7.5.

7.1 Problem Definition: Auto-pilot Mode in Interactive Process Modeling

In Chapter 1, we described Goal 2 of interactive process modeling/discovery, using several challenges/subgoals associated with it. The subgoals dealing with structural representations of process models, and guarantees regarding soundness were addressed in Chapter 3 and Chapter 4 by using free-choice workflow nets and synthesis rules. Furthermore, in Chapter 5 and Chapter 6 we addressed the subgoals to enable interactive modeling of labeled free-choice workflow nets, supported by the information from the event log, using projections and fast conformance analysis. In this chapter, the focus is to move further towards automated discovery of process models, by keeping the human-in-the-loop, i.e., to address the subgoal of providing recommendations. With this as the aim, we address the following problem in this chapter:

- How can we assist the user in semi-automated interactive process discovery, such that:
  - multiple recommendations for positioning an activity in a process model can be automatically generated and presented to the user. Until now, the user had to manually add an activity at the desired location in the process model. However, it would be desirable to automatically place an activity in a process model. Moreover, the user should ideally be able to choose from multiple (automatically generated) recommendations for positioning an activity in the process model. Suggesting the position of an activity in a process model without overwhelming the user with too much information is one of the foremost challenges of human-in-the-loop process discovery techniques. The selected activity could be placed at multiple positions within the process model. Therefore, it is vital to recommend to the user the most *relevant* locations where a particular activity can be placed in the process model. Eventually, the user needs to judge the appropriateness of a particular recommendation based on the comprehensibility of the process model for a given recommendation, along with other quality metrics. Moreover, it would also be ideal to provide the user with *the insights into each recommendation, i.e., how does the recommended positioning of an activity compare with the information from the event log*. Ideally, the recommendations should be ranked automatically based on the quality dimensions computed from the event log, in order to assist the user in decision making. Therefore, there is a clear need to automatically calculate multiple recommendations for adding a particular activity to a process model, and to automatically rank these recommendations, to assist the user in decision making.
  - the recommendations about the next activity that should be added to the model can be automatically generated and presented to the user. There should be a certain intuitive *flow* of choosing the activities during the process of process model construction. Since the process model is expanded incrementally by adding one activity at a time, there should be a logical order in which the activities are added to the net. That is, there should be a way to decide the order, or decide which activities belong together. For example, from a user's perspective, it would be easier to get recommendations of all the related activities together (or following one another), rather than getting ad-hoc recommendations of unrelated activities. Furthermore, at any given point during process discovery, the user should be able to alter the logic in which the next possible activity recommendation is generated.
  - the user can switch between manual process modeling and automated process discovery. It is important to allow the user to switch effortlessly between automated discovery and interactive discovery. The user may decide to build a process model until a certain point, and then wish to *hand it over* to a discovery algorithm to automatically complete the rest of the

process model. That is, a user's focus may be to exhaustively discover a process model by using the available domain knowledge first, and then let an automated technique take over. Alternatively, the user may want to delegate some discovery tasks to an automated discovery algorithm, and then resume interactive process discovery when the so-called *quality* of the process model falls below acceptable levels.

In the next section, we discuss the proposed solution to address the problems stated above. In particular, we address the problems discussed above in the context of labeled free-choice workflow nets, as it is the class of process models that we have used in to enable interactive process modeling as shown in Chapter 3 to 5.

7.2 Enabling Semi-automated Process Discovery

In this section, we discuss how we can address the goals discussed above, in order to enable semi-automated process discovery. The design choices of the system have been inspired from [43]. Figure 7.2 shows the high level overview of the main components

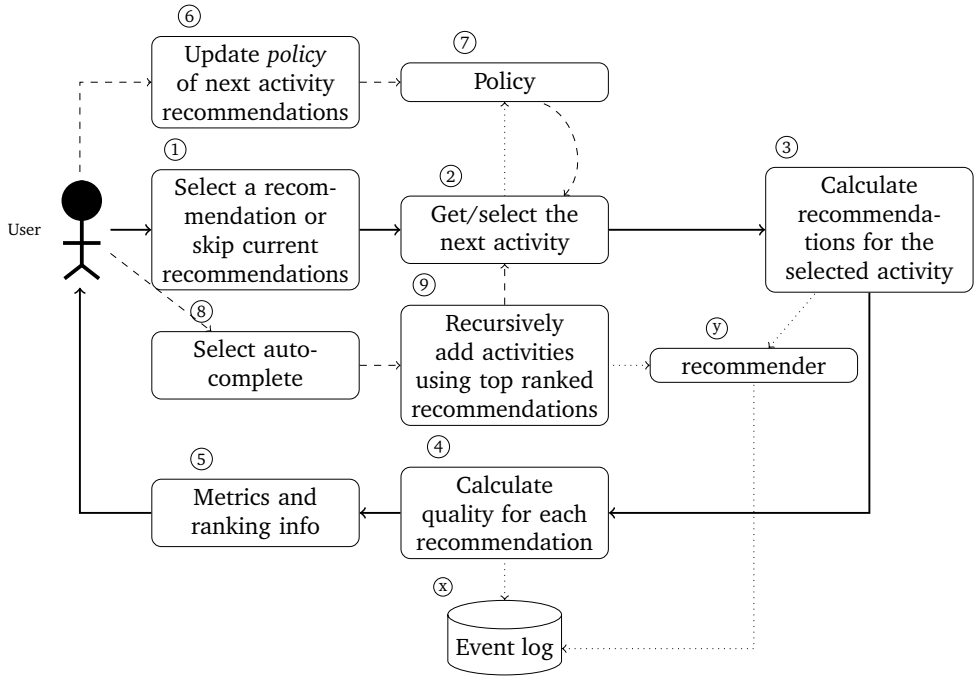


Figure 7.2: Components diagram of the proposed solution. Thick lines indicate the usual flow and interactions between the various components. Dashed lines indicate alternative flow and interactions between components. A dotted line between two components from A to B indicates that component A uses component B.

of the proposed solution. The usual flow and interactions between the components of Figure 7.2 is as follows: the user selects one (or skips all) recommendation(s) from the list of recommendations ①, the process model is updated based on the recommendation chosen by the user and the next candidate activities are selected ② depending on the policy chosen ⑦. The information from the event log ⓧ is used by a recommendation algorithm ⓨ to come up with different recommendations for the newly selected activity/activities ③. The quality of each recommendation is calculated ④ by using the event log ⓧ which is then presented to the user along with ranking information ⑤. At any point of time, the user can change the policy used to predict the next activity/activities ⑥. Similarly, the user can decide to hand over the discovery task to the automated discovery technique at any point of time ⑧ and ⑨. The auto-complete mechanism recursively adds activities to the process model by using the recommendation algorithm ⓨ by selecting the top ranked recommendation until some termination condition is met. The flow resumes with Get/select the next activity ② after performing ⑥ and ⑦ or ⑧ and ⑨. In the following sections, we discuss the way in which the proposed solution has been realized.

7.2.1 Next Possible Activity Recommendation

In this section, we discuss the strategies to decide the next activity that could be added to a labeled free-choice workflow net. Since process discovery in our approach is incremental in nature, the order in which the activities are added to the labeled free-choice workflow net is important. There are a couple of reasons for this. Firstly, having a logical flow of adding activities incrementally may assist the user in better decision making. For example, adding all the semantically similar activities one after the other may help the user in structuring the overall labeled free-choice workflow net better than adding unrelated activities in random order. Secondly, by following a strict order, it may become possible to discover a certain structure of labeled free-choice workflow net which may otherwise become unreachable because of prior discovery decisions when adding activities randomly. The policy is a pluggable component and there could be many policies used to decide the next possible activities that are added to a labeled free-choice workflow net. At any given point, the user can switch between the policies by updating the policy using component ⑥ of Figure 7.2. Here we present two policies which suggest a *single* next activity for a labeled free-choice workflow net.

Alphabetical

The *alphabetical* policy, as the name suggests, orders activities purely based on their alphabetical ordering. At any point, if a user decides to skip an activity, then the next activity based on the alphabetical ordering is chosen. As evident, in the case of alphabetical ordering, only one activity is selected at a time. Even though this is a very simple policy, it can still be useful as the user *knows* which activity to expect next.

Log Heuristics

The *Log heuristics* based policy orders activities based on the information extracted from the event log. The idea is that the user starts constructing the labeled free-choice workflow net with the most common starting activities across all the cases from the event log, and ending it with the most common ending activities across all cases in the event log. The activities in-between are also ordered according to their occurrences across the cases in the event log. This policy is described in more detail below, but in order to do so, we first introduce and revisit some notations. In essence, we assume that a user would typically construct process models in a sequential way, starting with the most common first activity across all the cases, and proceeding towards the most common end activity across all the cases.

Let L denote an event log. We know that a case is a sequence of activities. It should be noted that the same activity can be repeated multiple times in a case, and hence can be present at multiple positions in the activity sequence of a case.

Let $\mathscr{A}$ denote the set of all the activities from the event log L. Then for any activity $a \in \mathscr{A}$, let $\#_i(a)$ denote the *total number of cases* for which the activity at the i^{th} position is a.

In order to compute the policy of log heuristics, we introduce a function $max(act)$. $max(act)$ returns "the" position where activity act occurs most frequently across all the cases from the event log. In case there are multiple positions with the highest frequency, then the lowest position is chosen. Let $a, b \in \mathscr{A}$ be two activities, then we define a partial order $a \leq b$ as:

$$a \leq b \equiv max(a) < max(b) \text{ or}$$
$$(max(a) = max(b) \wedge \#_{max(a)}(a) \geq \#_{max(b)}(b))$$

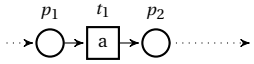
$a \leq b$ defines a partial order. In case $a \leq b \wedge b \leq a$, then this order is defined randomly between a and b. The partial order can be made total to form a sequence. This sequence forms the policy of log heuristics, that suggests the position of an activity based on the information from the event log, across all the cases and compared to all the other activities.

Clearly, more policies could be devised based on various other factors. For example, the user may be interested in adding activities in *reverse sequence*. That is, the user may be interested in starting with the most common final activity, and gradually move towards the starting activities in a process model. Such a policy would essentially present the activities in reverse of the log heuristics policy proposed above. Another interesting variant of the policy could be to use the most frequent activities in the event log first, in order to model the so-called *backbone* of the process model. And then eventually add infrequent activities one-by-one. The policy component in itself is expendable, and can thus be extended with more such policies.

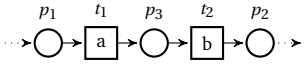
7.2.2 Generating and Evaluating Recommendations

The *recommendations* component provides a list of possible edits that can be made to the current labeled free-choice workflow net, to incorporate the next activity as

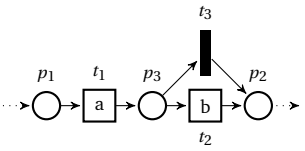
suggested by the policy. We use an approach based on synthesis rule patterns for populating possible ways of adding an activity to a labeled free-choice workflow net.



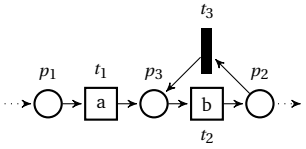
(a) Example fragment of a labeled free-choice workflow net. The activity a in the net occurs directly before activity b - which should be added next.



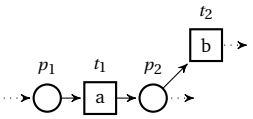
(b) Adding b using ψ_A^{WF} rule. This could result in multiple labeled free-choice workflow nets, depending on the different possible applications of ψ_A^{WF}.



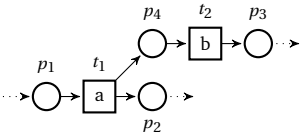
(c) Adding b using ψ_A^{WF} rule, followed by adding a silent transition using ψ_T^{WF} rule.



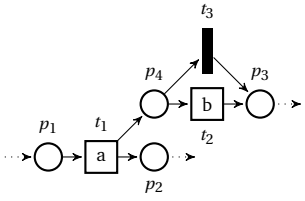
(d) Adding b using ψ_A^{WF} rule, followed by adding a silent transition using ψ_T^{WF} rule. It should be noted that the input to transition t_3 can contain additional places other than p_2 to satisfy the free-choice condition.



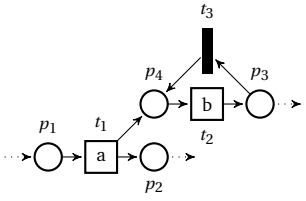
(e) Adding b using ψ_T^{WF} rule. This could result in multiple labeled free-choice workflow nets, such that each application of ψ_T^{WF} contains p_2.



(f) Adding p_4 using ψ_P^{WF} rule, followed by adding b using the ψ_A^{WF} rule. p_4 can be added in multiple ways, using all the possible applications of ψ_P^{WF} rules with t_1 (along with some other possible transitions) as input.



(g) Adding p_4 using ψ_P^{WF} rule, followed by adding b using tue ψ_A^{WF} rule. Finally adding a silent transition t_3 using ψ_T^{WF}.



(h) Adding p_4 using ψ_P^{WF} rule, followed by adding b using the ψ_A^{WF} rule. Finally, adding a silent transition t_3 using ψ_T^{WF}. The input of t_3 can contain additional places other than p_3 to satisfy the free-choice condition.

Figure 7.3: Example of patterns based on multiple applications of synthesis rules in order to develop recommendations to add an activity b (which is directly preceded by a).

Essentially, this results in multiple candidate labeled free-choice workflow nets, of which each labeled free-choice workflow net contains a different way of positioning the desired activity. The procedure followed for populating the candidate labeled free-choice workflow nets is as follows:

1. For a given labeled free-choice workflow net and an activity b chosen to be added next, find out the activities from the labeled free-choice workflow net which happen directly before the activity to be added next, according to the filtered event log. The event log is filtered to contain only those activities which are present in the labeled free-choice workflow net and the activity to be added next.

2. For each activity a from the labeled free-choice workflow net satisfying the above condition (i.e., directly occurring before b), use all the patterns from Figure 7.3, to add the activity to be added next. Usage of each pattern results in a different labeled free-choice workflow net, which could serve as a recommendation. In case multiple transitions in the labeled free-choice workflow net are labeled with activity a, then use the patterns from Figure 7.3 corresponding to each such transition. Furthermore, the patterns from Figure 7.3 are also used corresponding to each silent transition from the labeled free-choice workflow net.

It should be noted that we do not claim that the patterns from Figure 7.3 are complete. At any given point, the structure of the labeled free-choice workflow net dictates the types of labeled free-choice workflow nets discoverable using Figure 7.3. Hence, depending on the sequence in which activities are added, and the prior decisions made (patterns chosen), it may no longer be possible to deduce certain structures. For example, consider the care-pathway process from Figure 7.4. Let us assume that this is a labeled free-choice workflow net (can be easily done by adding two silent transitions $(\top, \bot)$ and two places after i and before o). Suppose we have to add an activity *Re-Diagnose* in the loop, before the activity *Diagnosis 2* (t_{13}), to obtain the process model similar to the one shown Figure 3.1. However, none of the patterns from Figure 7.3 would support this structure. In such circumstances, the user may want to switch to manually editing the labeled free-choice workflow net using the approach from Chapter 5 in order to first rename t_{13} with *Re-Diagnose*, and then add *Diagnosis 2* in sequence.

For every possible position of the activity in the process model, we then compute the fitness and precision scores in the changed labeled free-choice workflow net and the event log, using the conformance technique discussed in Chapter 6. Each recommendation has its own fitness and precision score and hence indicates the goodness of positioning an activity at various locations in the process model. It should be noted that the fitness and precision values are calculated with respect to the activities present in the labeled free-choice workflow net. It is clear that the newly added activity can be added in multiple ways. Therefore, the precision or fitness scores would vary depending on the different ways in which the new activity is added. The recommendations are *ranked* based on a ranking criterion, consisting of a weighted

average of fitness and precision values. By default the precision and fitness values are weighted equally, but these weights can also be determined by the user at any given point. To avoid an overload of information, the user can also set filters to only show recommendations above a certain threshold.

7.2.3 Mix and Match - Auto Discovery and Interactive Discovery

The *auto-discovery* component enables the possibility of using automated process discovery along with interactive process modeling. This component allows the user to pass the control to the recommender component. The user provides some input to

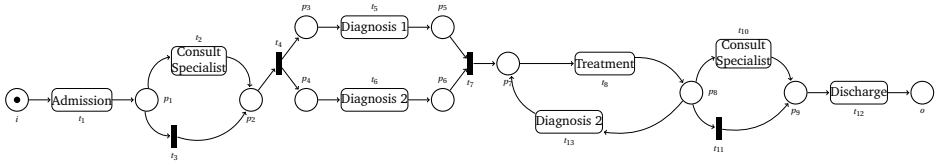


Figure 7.4: An example process model from Figure 3.1, without the Re-Diagnose Activity.

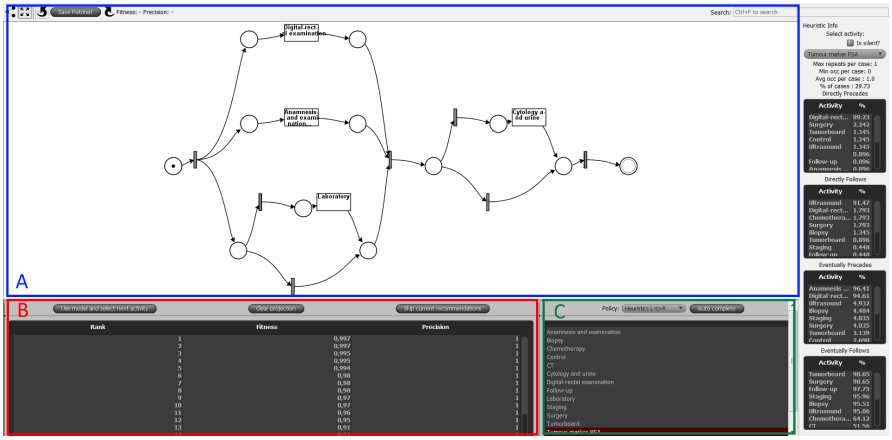


Figure 7.5: User interface of the proposed approach: the process model view (A) shows the labeled free-choice workflow net interactively discovered by the user, the recommendations table (B) shows the ranked recommendations in a tabular form, the policy and activity selection view (C) shows the activities that can be added to the process model, and the information abstracted from the event log on the right side (from Chapter 5).

the algorithm, similar to a traditional automated discovery technique which requires some pre-configuration. The user sets a threshold for minimum fitness and precision values. The user can also update the weights for fitness and precision scores used for ranking the recommendations. Once these parameters are set, the auto-discovery feature iteratively adds activities to the labeled free-choice workflow net, while the thresholds are met, by choosing the first ranked labeled free-choice workflow net from the list of recommendations. The labeled free-choice workflow nets of the activities which did not satisfy the threshold criteria are skipped. Finally, the user can resume the interactive process discovery. This component allows the user to switch between automated and interactive discovery and can be utilized in multiple ways. For example, the user may let the algorithm discover a high-level labeled free-choice workflow net that is clearly evident from the data and then add the intricate details based on the domain knowledge. Alternatively, the user may interactively discover the labeled free-choice workflow net up until a certain point and then delegate the discovery of the remaining activities to be computed automatically.

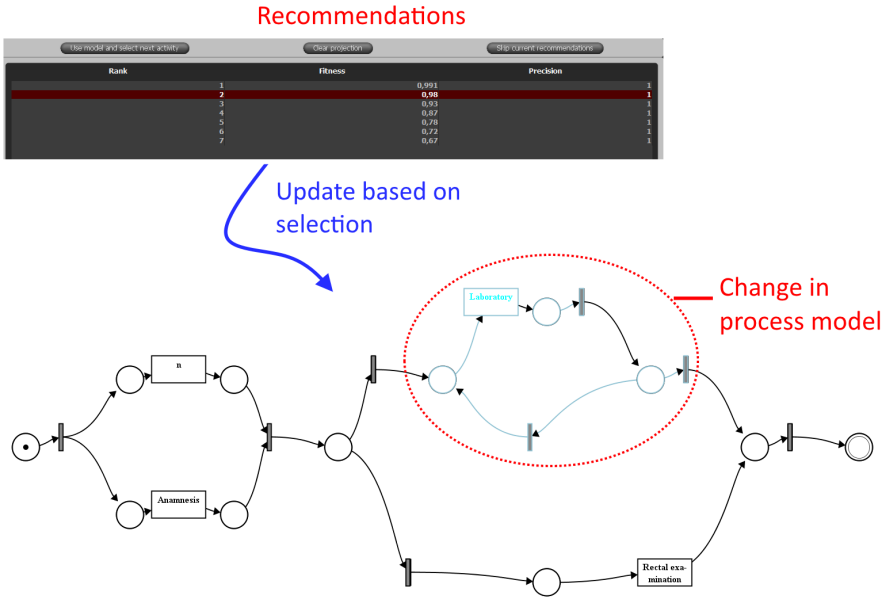


Figure 7.6: Based on the recommendation chosen from the recommendations table, the labeled free-choice workflow net is updated temporarily to display the change in the labeled free-choice workflow net.

7.3 Interface and Interaction

The proposed solution is composed of three primary display and interaction panels as shown in Figure 7.5. In this section, we describe the user interface of each panel, and the type of interactions possible that enable human-in-the-loop process discovery.

7.3.1 Process Model Panel

The *process model panel* is the panel **A** from Figure 7.5. Whenever a user chooses a recommendation, the labeled free-choice workflow net in this panel is updated by placing the new activity in the net based on the recommendation. Along with the process model view, there is an option to *undo* a previously selected recommendation, thereby allowing the user to revert the changes made. Of course, the user always has the option of directly editing the labeled free-choice workflow net based on the synthesis engine as shown in Chapter 5.

7.3.2 Recommendations Panel

The *recommendations panel* shows the recommendations of possible edits to the labeled free-choice workflow net based on the chosen activity. This is a tabular panel, corresponding to panel **B** from Figure 7.5. Each row in the table corresponds to a unique way of adding an activity to the current process model. The table has three columns: *rank*, *fitness* and *precision*, that guide the user through the impact of each recommendation. Figure 7.6 shows the primary user interaction of the proposed approach. Based on the ranking, fitness and precision scores, the user selects one row from the recommendations table. The change is animated and projected on top of the current view of the labeled free-choice workflow net. The layout is changed minimally, and the new nodes added to the labeled free-choice workflow net are colored differently, so that the user can easily spot the part of the net that has changed. By navigating through different rows (that is recommendations) the user can readily see the impact of each change. If a user is satisfied with a particular recommendation, then that recommendation can be made permanent by clicking *Use model and select next activity* button. This button finalizes a recommendation, chooses the next activity based on the policy and generates new recommendations based on the next activity chosen. Alternatively, the user may click the *Clear projection* button to clear the current recommended projection from the labeled free-choice workflow net or click the *Skip current recommendations* button to skip all of the given recommendations, choose a next activity based on the policy and generate new recommendations based on the newly chosen activity.

7.3.3 Policy Panel

The *policy panel*, as shown in Figure 7.7, corresponds to panel **C** from Figure 7.5. This panel provides the user with options for choosing or updating the policy to select the next activity. The policy panel contains a policy selection box which contains all

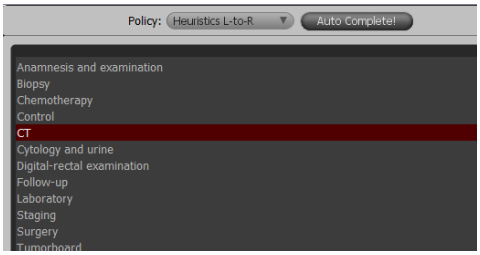


Figure 7.7: Panel showing the chosen policy, Auto-complete button and activities.

the available policies. The policy panel also contains a list box for activities which contains the activities from the event log. The user can scroll through the activities. The current activity as chosen by the policy is highlighted. Furthermore, the activities in the activity list are sorted according to the chosen policy. This provides an easy way for the user to find out the sequence of activities as suggested by the policy. The user can also interact and select any activity from the activity box. This action overrides the policy and gives the user more control over the sequence in which activities should be added to the labeled free-choice workflow net.

Whenever a new activity is chosen, either automatically by a policy, or manually by the user, the recommendations are recalculated for the newly selected activity. The policy panel also has the *Auto-complete!* button. As discussed previously, the auto-complete functionality lets the user switch between interactive discovery and automated discovery. Upon clicking the auto-complete button, the user is provided with a dialog box to set the thresholds for minimum fitness and minimum precision values, as well as the weights for fitness and precision values that should be used for ranking.

7.3.4 Design Evolution

The overall design of the system went through a number of iterations. Finally, the system ended up with the three primary sub-components influenced by various methods from the literature, as discussed earlier. The idea behind designing the tool decomposed into three main sub-components is to clearly separate the steps of the workflow in the usage of the system. The process model is the core component of process discovery and hence is the largest component and is placed centrally. The recommendations panel contains the possible recommendations for changes in the process model. This is the *first* step of the workflow, wherein the user goes through the recommendations and chooses one. Since it is natural to navigate (or read through) a workflow from left-to-right, this panel is placed on the left-hand side of the tool. The policy panel shows all the activities and highlights the next activity that is chosen. It should be noted that the next activity is typically automatically chosen and highlighted based on the policy chosen. Hence this is typically the second step of the workflow, and so this panel is placed on the right-side of the recommendations panel.

Having discussed the idea behind the overall design, and positioning of various

panels, we now briefly discuss the design decisions made within some of the panels. The policy panel (panel **C** from Figure 7.5) shows the activities as a list. The idea behind this is to let the user freely navigate through different activities, and check the natural sequence of following or preceding activities based on the chosen policy. The design of the recommendations panel underwent multiple changes. Initially, the recommendations panel was designed as *tabs*, wherein each tab contained a new process model based on the recommendation. However, it was difficult for the user to comprehend the impact of a change in the process model. Hence, after exploring multiple design ideas, it was decided to show the changes in the process model projected on the current process model. Also, by using different colors, the user can easily view the impact and the change in the process model. Furthermore, it was decided to provide the recommendations in a tabular format. The tabular design was chosen to provide a holistic view that is easy to sort and navigate, showing all the recommendations and their corresponding quality scores.

7.4 Implementation and User Study

The proposed approach is supported by the nightly build version of ProM [171], in the *InteractiveProcessMiningLite 6.9.13* package, under the plug-in "Petri net editor with predictor view (ProDiGy)". Of course, this plug-in also contains the functionality discussed in Chapter 5 and Chapter 6. This plug-in requires an event log in XES [3] format and an AHL (AggregatedLogHeuristics) object of the event log as input. The screenshot of the tool is shown in Figure 7.5.

In order to understand the usage, behavior and implications of the proposed solution we performed a user study wherein domain experts explored the system based on some tasks using a synthetic data set.

The overall goal of the study was to understand the usability of the proposed system and to gain insights through user feedback. The final output generated by the users using the system were also evaluated and compared with the traditional automated process discovery techniques.

The user study was conducted in the context of oncology patient flow in a hospital based on simulated data. We used Figure 2 in [181], as recreated in Figure 7.8, as our reference process model. This process model contains the expected workflow of patients suffering from three types of cancer: prostate, bladder and kidney. This process model was used to simulate the data of 850 patients: 250 each of prostate and kidney cancer, and 350 of bladder cancer. Furthermore, in order to replicate the reality, noise was introduced in the data by removing 3% of random activity occurrences from random locations and adding 3% random activity occurrences at random locations in the event log. This 'noisy' event log was used throughout the user study. Three industry based health-care researchers participated in this study. As a first step, all the participants were provided with a brief introduction of the tool. Next, the participants were asked to perform a task of discovering an end-to-end process model for all the patients using the entire (noisy) event log. The above task was used to guide the domain experts in exploration of the proposed system, and their feedback

was actively registered during the usage of the tool via unstructured interviews. Furthermore, after their usage of the tool, the users were presented with a questionnaire to analyze the things that could be improved, were interesting and missing.

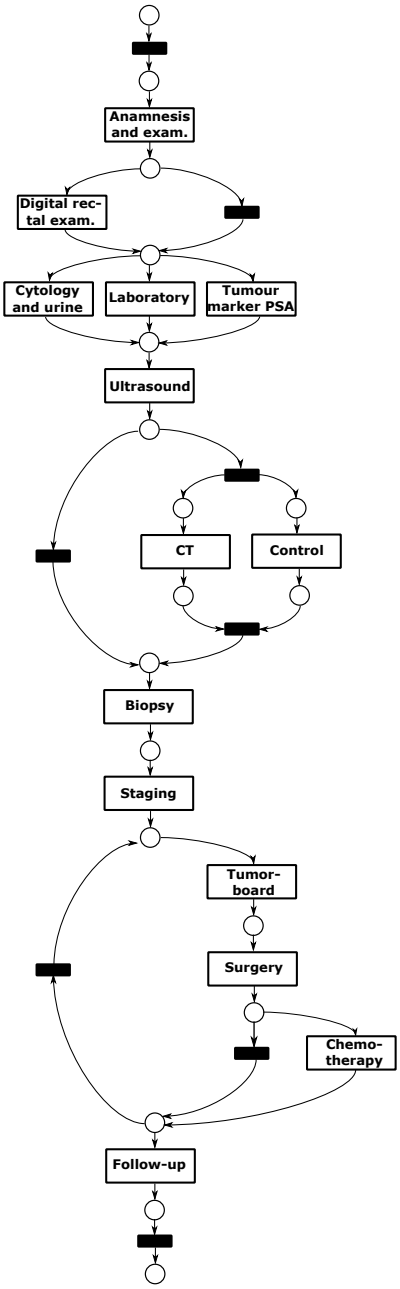


Figure 7.8: Example of an oncology patient workflow (Original model).

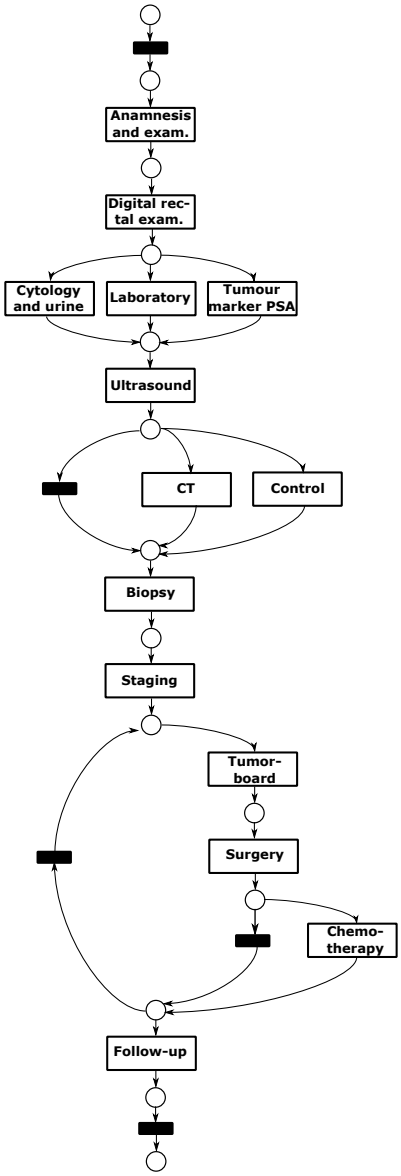


Figure 7.9: Process model discovered using our approach by participant **P1**. This model is almost similar to the original model from Figure 7.8. However, here the activity *Digital rectal exam.* is mandatory, whereas in the original model it could be skipped. Also, in the original model, both the activities CT and $Control$ can occur. However, in this process model, only one of these two activities can occur.

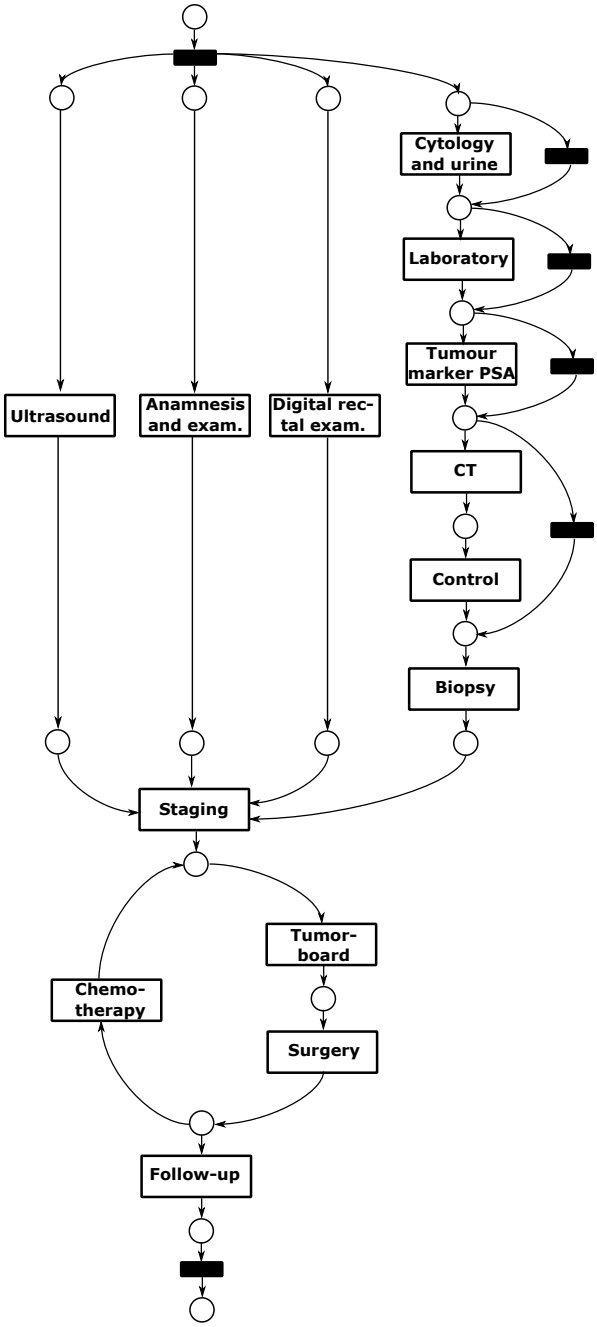


Figure 7.10: Process model discovered using our approach by participant **P2**. This model has some differences compared to the original model. For example, in the original model, the activities *Anamnesis and exam.*, *Digital rectal exam.* and *Ultrasound* are all in sequence, whereas in this model, they are in parallel.

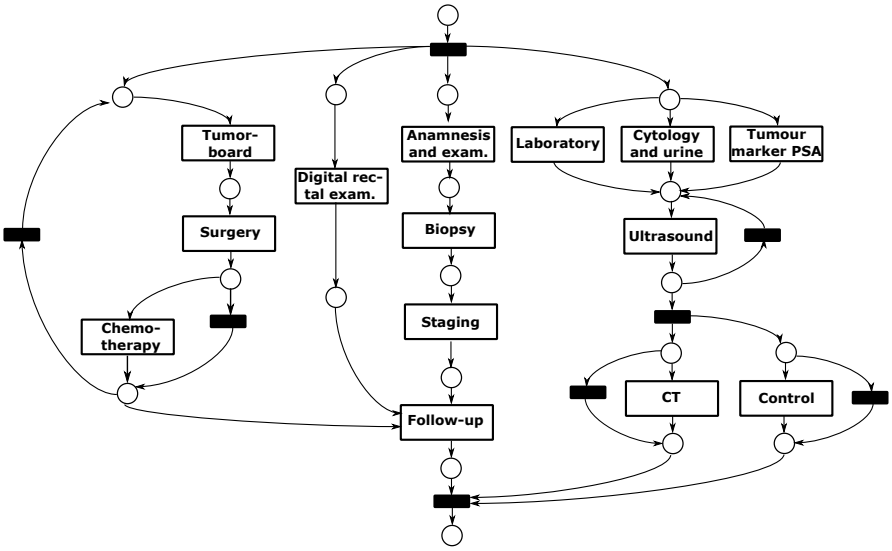


Figure 7.11: Process model discovered using our approach by participant **P3**. This model has a lot of differences compared to the original model. In general, this model contains many activities which are placed in parallel, compared to the original model.

7.4.1 Results

In this section, we discuss the key findings of the user evaluation. For convenience, the participants are labeled as **P1**, **P2** and **P3**. The process models discovered by the three participants are shown in Figure 7.9, Figure 7.10 and Figure 7.11 respectively.

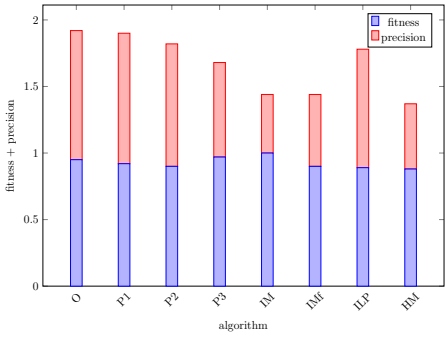
7.4.2 Usage Patterns

Even though every expert was given a standard introduction to the tool, there were differences in the way the tool was used. **P1** relied almost entirely on the ranking information noting that he *"trusts the data more"*, and therefore he always chose either the first or second ranked recommendation from the recommendations list. Contrary to this, **P2** used his domain knowledge most of the time, and relied on the recommendations when he was *"unsure what to do, or where to add an activity"*. **P3** added the activities he was familiar with first, and then relied on the recommendations to decide the fate of unfamiliar activities. The users were satisfied with the features and the workflow of the tool, and they deemed the system easy to learn. There were suggestions made by the participants based on their specific way of discovering a process model. For example, **P2** who relied more on his domain knowledge, navigated back throughout the interactive process discovery session using the *undo* button, and said that he *"misses a redo button"* to navigate forward. Similarly, **P1** who relied more on the information from the event log noted that *"the recommendations helped in guiding the process discovery, however sometimes the ranking difference between recommendations was hard to spot due to a very small difference in fitness and precision values"*. In most of the intermediary steps, the participants chose one of the top-3 recommendations. Also, none of the participants selected a recommendation ranked lower than 5 at any given point. The proposed system supported both the types of users sufficiently: the ones relying on using the domain knowledge as well as the ones relying on using the information from the event logs for process discovery, thereby supporting multiple ways of discovering process models.

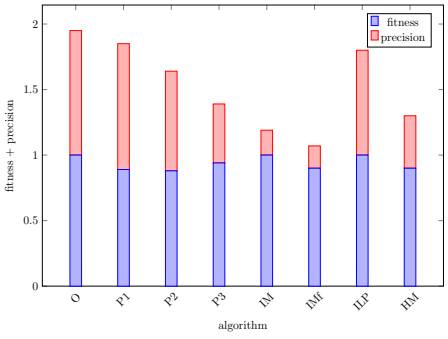
7.4.3 Quality of the Discovered Models

All three participants were satisfied with the process models discovered by them using our tool. The comparison of fitness and precision scores for the process models discovered by the participants, the original model, and some automatically discovered process models is shown in Figure 7.12. In Figure 7.12, we are interested in primarily comparing the overall quality of process models given by a combination of fitness and precision scores. Clearly, there are some notable differences between the original process model and the process models discovered by the three participants. For example, the process model modeled by **P1** does not allow skipping of the activity *Digital rectal exam*. The drop in the fitness and precision values between the noisy event log and the original event log could thus be attributed to such differences in the process models. However, in most of the cases, the process models discovered by the experts performed better than the automated process discovery techniques.

In general, the process models discovered by the experts were deemed simpler to understand and/or more appropriate by the respective experts, compared to the process models discovered by the automated techniques. As **P1** responded about one of the process models discovered automatically, *"it is extremely complicated and doesn't make much sense, the interactively discovered process model is easier to interpret"*. All the participants agreed that the proposed system enabled them to have more control over process discovery and suggested that the interactive process discovery technique enabled discovery of substantially better process models, especially in terms of intelligibility, compared to the traditional automated process discovery techniques. An interesting observation here was that even though the three participants started off with the same initial labeled free-choice workflow net and with the same event log, there were notable structural differences in the final labeled free-choice workflow nets. Partly this could be attributed to the usage patterns. However, another reason for these differences could be the fact that each participant had specific preferences to certain structural constructs in the labeled free-choice workflow nets over the others. For example, **P1** preferred sequential constructs over concurrency, whereas **P2** and **P3** had no such preference. This is also reflected in Figure 7.12, in which the labeled free-choice workflow nets discovered by almost all the participants have a similar fitness score, however, there is a notable difference in the precision score of each labeled free-choice workflow net. In essence, the process models discovered by the three participants are comparable with the baseline.



(a) Process models evaluated against noisy event log, i.e., the one which was used to discover the process model.



(b) Process models evaluated against the original noise-free event log.

Figure 7.12: Fitness and precision scores of the process models discovered from the noisy event log by three participants (P1, P2, P3) along with the automated discovery algorithms: Inductive miner (IM), Inductive miner infrequent (IMf), ILP Miner (ILP), Heuristics miner (HM). The original model (O) serves as the baseline.

7.4.4 Experts Gained Insights during Process Discovery

The strength of the traditional process discovery techniques lies in understanding the data to make certain decisions. But the decisions made during process discovery are not visible to the end user. Since our tool enables active user involvement in process discovery, it also leads to exploring intermediary patterns, which may otherwise remain unexplored or hidden. For example, the information about the link between the activities which might not be directly connected directly in the final process model. However, such activities may be directly connected during the intermediate steps of interactive process discovery. Hence, by encouraging the users to interactively discover a process model, the proposed system helped in gathering insights during the process of process discovery, as noted by **P3**: *"the system is surprisingly good for exploratory purposes"*.

7.4.5 Limitations of Our Study

The overall feedback by all three participants was positive and led to a lot of suggestions and possibilities for improvements. Nevertheless, we are aware that the user study has some limitations. Firstly, even though the participants had more than 50 years of health-care experience between them, the number of participants is limited and hence the results may be biased. Due to the sample size of the study, we did not conduct structural interviews and statistical analysis of the results. The participants from the current study also had some basic exposure to the field of process mining, whereas additional training may be needed for other populations.

7.5 Conclusion and Future Work

In this chapter, we presented a novel approach for providing recommendations to guide interactive process modeling. We first identified the key challenges of recommendations based interactive process discovery and proposed solutions in order to address these challenges. The solutions proposed were designed and implemented as a plug-in in ProM, and was evaluated by users from the health-care domain. Our evaluation demonstrates that the proposed system can outperform traditional automated process discovery techniques. Furthermore, the user study also demonstrated that the tool is easy to learn and the visual analytics techniques incorporated in the tool are intuitive. We have improved the usability of the system based on the feedback given by the participants of the user study. For example, we have added a new button that allows the users to "redo" a change. Furthermore, we have also added a new metric called log coverage, which provides the information about the percentage of the log covered. This metric complements the metric of fitness and precision, so that the users can have a better understanding of the prominent activities in the event log, compared to the infrequent activities.

Finally, we would like to provide some future directions in order to improve the proposed system. As noted already, the users may find it difficult to distinguish bet-

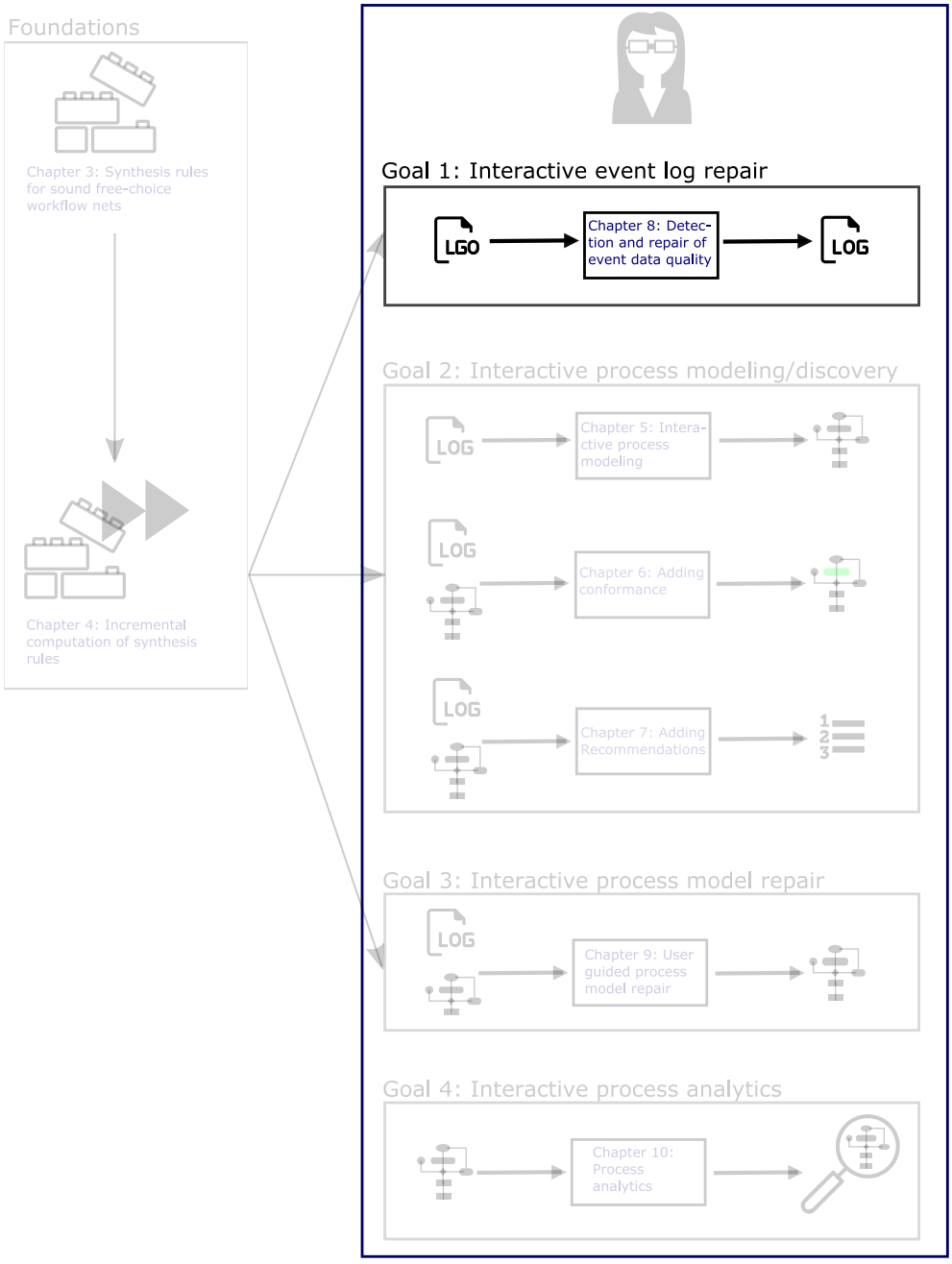
ween different recommendations purely based on the fitness and precision values. It would therefore be ideal to include a collection of new metrics to guide the user in further distinguishing the recommendations suggested, especially when the differences between the fitness and/or precision scores are not significant. Furthermore, a domain specialist may find the metrics of fitness and precision to be very technical. Hence, it would also be interesting to include some domain specific metrics (such as certain protocols that must be followed), that assist the user in choosing better process models from the list of recommendations.

One more future direction could be to change the way in which process models are constructed. In our approach, the process models are constructed by adding one activity at a time, by using simple patterns based on the synthesis rules. However, some users may prefer adding several activities at once. Hence, it would be ideal if the recommendation system can support recommendations of process model constructs by adding multiple activities at once.

Another interesting future direction could be constructing user profiles in order to build a recommendation system based on a user's background and preferences. The information from these user profiles could be used directly in the recommendation algorithm in order to populate only those process model structures, which are preferred by the user.

Chapter 8

Detection and Repair of Ordering Issues in Event Logs



This chapter is based on the publication [66].

The event log is central to all the process mining analysis techniques [163]. We know that each event is minimally characterized by attributes identifying the process instance to which it belongs (case id), the process step that was carried out (activity) and its timestamp. Events may optionally contain other attributes such as the person or machine that carried out the process step (resource).

As with other forms of data analysis, poor (event) data quality leads to poor analysis results (garbage-in, garbage-out). Most process mining tools are data-quality indifferent. That is, so long as the data can be parsed by the tool, its algorithm will process the data and generate output, thus requiring the analyst to conduct appropriate “reasonableness checks" when interpreting the results. While it is important to critically review analysis outcomes, systematically identifying and correcting data quality issues prior to analysis can increase the quality of the event log.

As discussed in Chapter 1, process mining techniques can be broadly categorized into process discovery, conformance analysis and performance analysis (enhancement). Process discovery aims to discover process models using the information from the event logs, either automatically, or by involving human-in-the-loop as discussed in Chapter 3–Chapter 7. Hence, it is vital to correctly sequence the activities within a case in order to discover accurate process models. Conformance analysis aim to determine whether activities as depicted in the event log were carried out in accordance with the organizational expectations as depicted by a process model, whereas performance analysis aim to determine whether activities and cases were executed differently. Hence, the ordering of events in an event log is central to all process mining analysis.

Timestamps are the principal means for ordering activities within cases and for calculating process execution milestones and performance. Hence, the quality of process mining analysis is highly contingent on the quality of timestamp data. For instance,

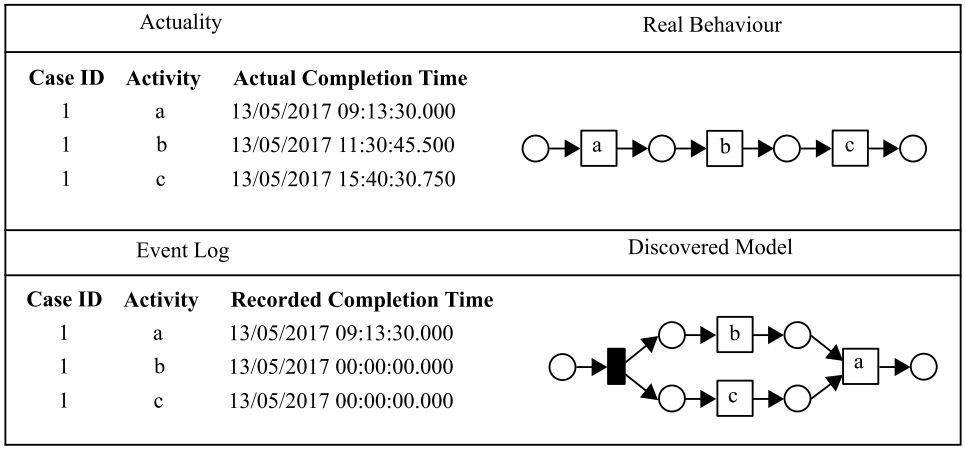


Figure 8.1: The effects on process discovery and performance analysis of inaccurate timestamps.

consider a situation, where activities in an event log are drawn from two different information systems, one of which records timestamps at only day level granularity, while the other records timestamps with millisecond accuracy (see Figure 8.1 for an example). The effects on process mining analysis, which can be quite dramatic, include discovered process models which do not reflect actual behavior and erroneous or misleading performance analyses.

In this chapter, we focus on addressing this concern of repairing timestamp issues in event logs by using domain knowledge. The remainder of this chapter is structured as follows. In Section 8.1, we discuss the main problem addressed in this chapter. In Section 8.2, we describe the “symptoms" of event ordering imperfection in event logs. In Section 8.3 we outline our approach to detect and repair timestamp quality issues, followed by details of detection, repair and impact of repair in Section 8.4, Section 8.5 and Section 8.6 respectively. We evaluate the proposed approach using two real-life event logs in Section 8.7. We compare our approach with the state-of-the-art in Section 8.8, and, finally, conclude in Section 8.9.

8.1 Problem Definition: Ordering Issues in an Event Log

The importance of timestamps in process mining analysis is well recognized [165] and a variety of authors have identified data quality issues affecting time-oriented data [93] and timestamps that impact on process mining analysis [21, 151], and others have suggested ways of characterizing the quality of available timestamp data. For instance, [120] characterize timestamps according to *granularity, directness of registration and correctness*. There exist tools for automated and user-driven detection and repair of time-oriented data, for example, TimeCleanser [92]. However, there are no techniques which provide consolidated detection and repair mechanisms to assist a user in dealing with time-ordering data quality issues in event logs in the context of process mining. This relates to Goal 1 from Chapter 1, which deals with interactive event log repair. Based on the three subgoals related to interactive event log repair discussed in Chapter 1, we formulate the following problem.

- How can we enable interactive event log repair to fix timestamp related issues using domain knowledge, such that:
  - a list of possible ordering-related data quality issues can be automatically generated and presented to the user. In order to repair an event log, the user must first be aware of the possible issues in the event log. The existing techniques from the literature [103, 153] typically provide a holistic view on the overall quality of the event log, by providing some global metric such as structuredness. However, in our case, we are interested only in the timestamp related issues in the event log. In particular, we are interested in finding out which activities possibly suffer from timestamp issues. Moreover, it would also be ideal to find out the type of timestamp issue that an activity is suffering from.

- the ordering-related data quality issues can be repaired by the user based on the domain knowledge. The simplest way of repairing timestamps of events could be using an excel like functionality, and repairing the timestamp related issues on a case by case basis. However, this is clearly not feasible in a practical scenario when the size of the event log and the number of events with timestamp issues is very high. Hence, it is desirable to allow the user to repair multiple instances of incorrect timestamps at once. In the literature, there exist techniques which automatically repair timestamps in an event log based on a pre-existing end-to-end process model [140, 141]. However, in reality, such a process model may not always exist. Instead, the user may only be aware of certain fragments of the process model. Hence, it would be ideal to effectively use the domain knowledge of the user in order to repair timestamps.
- an overview can be provided to the user to indicate the impact of the repair. That is, the technique should quantify the impact of repair actions on the event log so that the user can either decide keep the changes made or discard the changes made, where no significant benefit accrues through the repair.

In the next section, we set the scene by discussing what might cause ordering related issues in the event logs. This is followed by our approach to discuss detection and interactive repair of ordering quality issues in event logs.

8.2 Indicators of Event Ordering Imperfections

The approach proposed in this chapter aims to enable domain experts to detect and interactively repair event-ordering imperfections that arise due to timestamp-related issues. As a first step, we need to be able to *locate exactly where timestamp issues may exist in an event log* by recognizing characteristics commonly associated with event ordering imperfection. From existing literature that describes data quality issues in event logs [21, 119, 151, 163, 165] in particular, and time-oriented data in general [93], we abstract three classes of indicators that may be used to locate event ordering issues. It is not the intention here to provide a comprehensive list of indicators of event ordering imperfection; rather, our aim here is to highlight the importance of recognizing the various indicators of event ordering imperfections from an event log as a starting point for our event log repair approach.

8.2.1 Granularity

One of the indicators of event ordering imperfections is the existence of either coarse timestamp granularity (for example, *imprecise timestamp* [119, 163]) or mixed timestamp value granularity (for example, an event log that includes events from multiple systems where each system records timestamps differently [93, 165]). The mixture of varying timestamp granularity may result in events being ordered incorrectly. For

example, an event *'Seen by Doctor'* may be recorded at day-level granularity, for example, '05 Dec 2017 00:00:00'. Within the same case, another event *'Register Patient'* may have second-level granularity, for example, '05 Dec 2017 19:45:23'. The ordering of these two events will be *'Seen by Doctor'* followed by *'Register Patient'*, which is incorrect as it should have been the other way around. There are techniques for dealing with partially ordered event data, but we assume events to be totally ordered. Hence, the approach proposed in this chapter seeks to detect those activities/events with potential timestamp granularity issues to assist domain experts in making a well-targeted event log repair procedure.

8.2.2 Order Anomaly

Locating events affected by ordering imperfections can also be performed by identifying events exhibiting unusual temporal ordering, consider for example the duplicate entry of exactly the same event [93]. Given an activity a_1 that was recorded twice (due to a user mistakenly double-clicking a button on his/her screen), we may observe an unusual (incorrect) directly-followed temporal order $a_1 \rightarrow a_1$, highlighting a potential event ordering imperfection. Issues related to missing events [21, 119, 163] and incorrect timestamps due to events being recorded post-mortem [163] or due to manual entry [119] can also be detected by learning if there exist other forms of unusual temporal orderings between activities. The existence of infrequent temporal orderings between activities does not automatically mean that the event log has timestamp quality issues. However, the ability to highlight those activities will assist domain experts in deciding the best repair actions, if necessary.

8.2.3 Statistical Anomaly

Extracting more generic statistical anomalies, such as learning the temporal position of a particular activity in the context of other activities, or the distribution of timestamp values of all events in an event log, may indicate the existence of timestamp-related problems. For example, when an event log is comprised of events from multiple systems, there may be more than one way in which timestamps are formatted, which may lead to the 'misfielded' or 'unanchored' timestamp problem [93, 151], whereby timestamp values are interpreted incorrectly. A common situation is when timestamp values formatted as `DD/MM/YYYY` are interpreted as `MM/DD/YYYY`. In this situation, one may see imbalance distribution of the 'day of the month' values of all events in the event log whereby all events will have the date value between 1–12 only, and none for 13–31. There are other indicators of event ordering problems that can be detected by observing statistical anomalies, such as the use of batch processing [93, 151] and multiple timezone problem [93].

8.3 Approach

In this section, we provide a high-level overview of the proposed approach. We start with an event log and adopt an iterative approach to addressing event order imperfection. Figure 8.2 shows the steps involved in the process of event log repair.

As a first step, we perform automated, indicator-based issue detection in the event log. The way in which we detect potential event ordering imperfections is informed by the symptoms of the imperfection as described in Section 8.2. Based on the issues detected, a user, typically a domain expert, performs changes on the event log in order to correct some of the detected issues. The user is then provided with the impact of change made. Finally, the user may decide to keep the changes (i.e., replace the existing event log), or ignore the changes (i.e., go back to the prior event log). The cycle is repeated until the user is content with the quality of the event log.

In the sections to follow, we show through examples, how our indicators are used to detect potential event-order issues, how identified issues are repaired, using domain knowledge captured via process-fragments, and an alignment strategy. Finally, we describe a set of metrics for assessing the impact of the repair actions on the log which can guide the user in accepting or rejecting the repairs.

8.4 Detection

Having discussed the approach on a high level, we now focus on the first component of our system, i.e., automatically detecting time-oriented quality problems in the event log. Most of the techniques from the literature provide an aggregated measure of the quality of an event log, without pinpointing at the exact problems. In

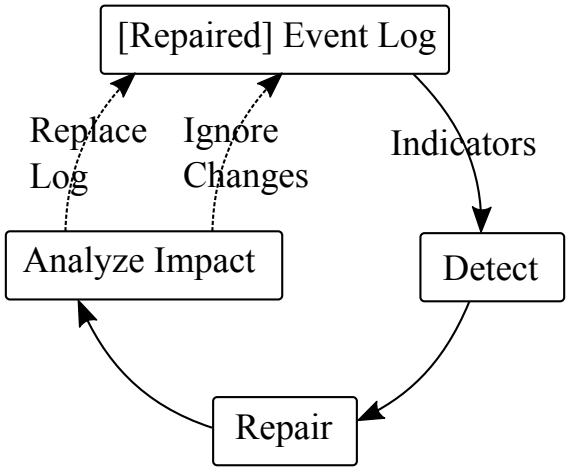


Figure 8.2: Approach to detecting and repairing event ordering imperfection

our approach, the user is presented with a consolidated list of possible time-related data quality issues, describing the possible problem, the affected activity (activities) and the number of instances affected. Also, unlike some of the traditional detection strategies, our technique doesn't require an end-to-end process model as an input for detection and the detection of issues is solely based on the information from the event log. In this section, we discuss the three detection strategies employed in our tool to detect the three symptoms presented earlier in Section 8.2.

8.4.1 Granularity-based Detection

The first detection mechanism deals with identifying the issues with the granularity of timestamps in the event log. In order to achieve this, we employ the most basic, yet quite powerful form of detection, i.e., inspecting the timestamp value of each event in the event log.

A table is maintained whose rows correspond to activities from the event log and columns corresponds to the granularity of timestamps. Thus each cell in the table represents the number of times the particular activity was recorded at the corresponding (highest) granularity in the event log. For example, if an event has the timestamp of '05 Dec 2017 19:45:00', then its granularity would be minute level, whereas if an event has the timestamp of '05 Dec 2017 19:45:13', then its granularity would be second level. We then identify the possible granularity issues for an activity by investigating the distributions over different granularities in the event log. Table 8.1 gives an example of such a table, which clearly indicates that activity *b* is usually recorded at a lower granularity (Hour) compared to the other activities.

8.4.2 Ordering-based Detection

The next detection mechanism deals with the ordering of events in the event log. Inspecting the log gives indications about possible granularity related issues in the event log. However, it does not provide much information about the *ordering* of activities within the event log.

Since there is no ground truth (process model) available, determining the correct order of activities is a challenging task. In a way, this task is exactly similar to the

Table 8.1: An example granularity table populated by scanning an event log.

Activity	Year	Month	Day	Hour	Minute	Second
a	0	0	0	1	11	489
b	0	0	0	450	20	0
c	0	0	0	0	5	350
d	0	0	0	2	7	448
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 8.2: An example *follows relations* table snapshot (left) and *precedes relations* table snapshot (right).

$\rightarrow$	a	b	c	d	e	f	$\emptyset$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
c	3	50	0	100	200	25	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$\leftarrow$	a	b	c	d	e	f	$\emptyset$
a	0	3	3	450	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
f	0	0	25	0	0	0	0

challenge faced by the process discovery techniques, which try to deduce the correct order of activities using an event log.

Inspired by how process discovery algorithms typically work [185], we use pairwise causal relations between activities to *guess* the correct order of activities based on frequency thresholds. In order to achieve this, we populate two tables: one containing the directly *follows* relations and the other containing the directly *precedes* relations between the activities in the event log. Each cell contains the number of times an activity from a corresponding row, was directly followed (preceded) by an activity from the corresponding column in the event log. For example, in the *follows* relations of Table 8.2, activity c is followed by activities a, d, e and f: 25, 150, 200 and 3 times respectively.

Having populated the *follows/precedes* relations tables, the next step is to analyze the discrepancies in the ordering with respect to each activity. We demonstrate this with the help of an example.

Consider the activity c in Table 8.2 and a (user-defined) threshold value of 0.8. The first step is to filter out the fewest activities (with descending frequencies) which are directly followed by c resulting in at least 80% of the total occurrences of c. In the follows table of Table 8.2, these are activities b, d and e. Next, for every non zero remaining activities (a and f) of the filtered Table 8.2, we check the corresponding precedes relations table with the threshold values. The precedes relations table is again filtered (with the threshold of 80%) to keep only the infrequent (non zero) activity relations. In the case of activity a, these would be activities c and b. Now, since both the directly follows relation $c \rightarrow a$ is infrequent, and the directly precedes relation $a \leftarrow c$ is infrequent, we conclude that there is an issue in ordering between activities a and c. On the other hand, for activity f, the activity c is within the threshold, that is, all the occurrences of f in the event log are preceded by c. Since activity f itself is highly infrequent (compared to c), the ordering between c and f is assumed to be correct.

Considering both the directly *follows* and *precedes* relations, we *detect* only those infrequencies that are ordering related. Similarly, eventually *follows* and *precedes* relations are used to explore long-term infrequent ordering relations. The eventually follows and precedes also result in two tables, similar to Table 8.2. The eventually follows value is first computed at the level of traces. The number of times activity x is eventually followed by activity y, is thus the total number of occurrences of y after

the first occurrence of activity x in the trace. Hence, in the eventually follows table, each cell contains the total number of times an activity from a corresponding row is eventually followed by an activity from the corresponding column across all the traces in the event log. The eventually precedes value is also computed at the level of traces. The number of times an activity x is eventually preceded by an activity y, is thus the total number of occurrences of y before the last occurrence of activity x in the trace. Again, aggregating the trace level information results in an eventually precedes table as shown in Table 8.2. These two tables are used in a similar way to directly follows and directly precedes ordering relations, in order to compute the possible long-term ordering issues in the event log.

8.4.3 Statistical

The last group of detection mechanisms relates to statistical anomalies. Admittedly, there are many forms of statistical anomaly that one could use to detect event ordering imperfection. In this chapter, we investigated one type of statistical anomaly that can be used to detect event ordering issues due to *misfielding* (see Section 8.2).

For instance, due to incorrect extraction of the data, the MM/DD/YY from the system might be incorrectly processed as the DD/MM/YY format in the event log. In this scenario, the highest *day* timestamp value for all affected events would be '12'. Chances are, if the highest day would be '12', the highest month would be ≥ '12'. In order to evaluate such issues, we employ a directional statistical technique- Kuipers test [106].

Kuipers test is used to test whether a given distribution is contradicted by evidence from a sample of the data. Kuipers test is especially useful in the scenarios when the data is circular as it provides cyclic invariance. That is, while analyzing the hourly distribution of activity distribution, timestamp values 23:59 hours, and 00:01 would be considered closer. For more details about the Kuipers test we refer the reader to [121] (pg. 99).

We employ Kuipers test to compare the timestamp distribution of each activity with all the other activities in the event log, at 6 levels: second of the minute, minute of the hour, hour of the day, day of the week, day of the month and month of the year. For example, we look at the distribution of an activity at the level of day of the month, and compare it with the distribution of all the other activities from the event log at the level of day of the month. The activities whose distribution is statistically significantly ($p<0.05$) different compared to all the other activities, would then show up in the detection list. An assumption here is that all the timestamps related to a single activity use the same format.

8.5 Repair

The insights gained from the detection phase are coupled with the domain knowledge in order to repair the event log. In the event log, we repair a single activity at a time. The repair workflow consists of four steps as shown in Figure 8.3. The first

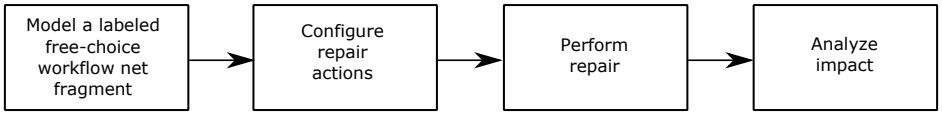
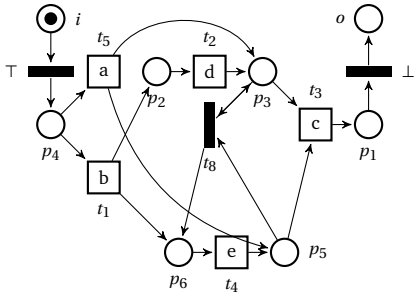


Figure 8.3: Approach for repairing event ordering imperfections.

three steps of Figure 8.3 are described in the subsections that follow. The last step is explained in the following section (Section 8.6). The idea is that, the user first models



(a) The expected (unknown) workflow.

Case	Activity	TimeStamp
1	e	1-1-2018 08:00
1	b	1-1-2018 09:00
1	d	1-1-2018 10:30
1	c	1-1-2018 11:00
2	a	2-1-2018 10:00
2	c	2-1-2018 11:00
2	e	2-1-2018 12:00
3	b	4-1-2018 09:00
3	d	4-1-2018 10:00
3	e	4-1-2018 10:10
3	e	4-1-2018 11:10
3	c	4-1-2018 12:00
4	b	5-1-2018 09:00
4	d	5-1-2018 10:00
4	c	5-1-2018 12:00

(b) Event log representing what really happened. Activity *e* is mispositioned or missing in some of the cases.

Figure 8.4: Expected process model and the real execution of the process depicted by the event log. Let us assume that the user is unaware of the expected process model.

a process model fragment, based on his/her domain knowledge, which contains the activity to be repaired along with some other activities. Next, the user configures some repair actions, in order to specify the timestamp value for the misaligned activity from the process model. Based on the chosen repair action, the timestamp value of the misaligned event is changed. Finally, the user is provided with an overview of the impact of the changes performed.

We use Figure 8.4 as the running example in order to explain the concepts discussed in this section. Let the process model from Figure 8.4a indicate the expected workflow. Clearly, the event log corresponding to this process model, shown in Figure 8.4b, has some ordering-related issues. Let us assume that the user is unaware and unfamilar with the complete end-to-end process shown in Figure 8.4a, and that the detection mechanism from Section 8.4 has indicated an timestamp ordering-related issue with activity *e*.

8.5.1 Modeling Process Fragments

The process of event log repair begins with the user specifying domain knowledge in terms of a labeled free-choice workflow net fragment using the interactive labeled free-choice workflow net editor discussed in Chapter 5.

Minimally, the labeled free-choice workflow net contains only the activity which should be repaired. The user can additionally add “other activities" to the labeled free-choice workflow net with respect to which the activity should be repaired. For example, consider that the expected model of Figure 8.4a is unknown. However, based on the insights gained from the approach in Section 8.4, the user knows that there is a possible issue in the ordering of activity *e* in the event log of Figure 8.4b. Let us assume that the user is aware of the relation between the activities *b, c* and *e*, and has modeled a process fragment shown in Figure 8.5.

It should be noted that in many real-life scenarios, the event log contains multiple activities and the user may not know the relation between *all* the activities. Hence it is ideal to allow the user to model partial (even long-term) fragments of labeled free-choice workflow nets, explicitly specifying the relations between activities that the user is aware of.

Modeling via labeled free-choice workflow net allows the possibility to include complex graphical fragments such as concurrency, choices, loops, duplications and introducing *silent* activities. Duplicate activities in the labeled free-choice workflow net are uniquely numbered. After modeling the labeled free-choice workflow net

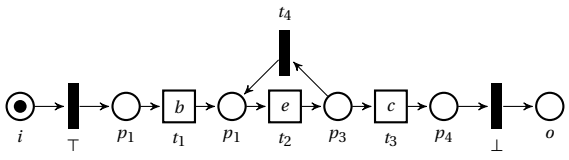


Figure 8.5: Fragment modeled by the user based on the available domain knowledge.

fragment, the next step is the configuration of the repair options.

8.5.2 Repair Configuration

Having designed a labeled free-choice workflow net fragment containing the activity to be repaired (among other activities), the user can then configure how the repair must be performed. The repair configuration allows the user to specify the settings for correcting the ordering of a particular activity, and is made of: (i) the repair activity configuration, and (ii) the repair action configuration.

The repair activity configuration deals with the activity to be repaired and is as follows:

- **Activity to repair:** Specify the activity to be repaired from the labeled free-choice workflow net fragment. For duplicate occurrences in the Petri net, select the (uniquely numbered) activity instance.
- **Add events?:** Specify if new activities should be artificially inserted in the event log. If set to true, the technique may insert artificial events (corresponding to the activity to be repaired) in the event log.
- **Remove events?:** Specify if activities should be removed from the event log. If set to true, the technique may remove existing events (corresponding to the activity to be repaired) from the event log.

The repair activity configuration is specified only once. For the example from Figure 8.5, let the repair activity configuration be as follows:

- **Activity to repair:** e (t_2)
- **Add events?:** *True*.
- **Remove events?:** *True*.

The repair action configurations describe multiple contexts for correcting the ordering of the activity, and consists of:

- **Anchor**: One activity from the labeled free-choice workflow net in relation to which the ordering should be corrected. This activity cannot be the same as the activity to be repaired. Alternatively, the anchor could also be the start of a case (first activity in the case) or the end of a case (last activity in the case).
- **Position** : Select whether the activity to be repaired should occur *before* the anchor or *after* the anchor.
- **Value** : Specify the value of the new timestamp of the activity to be repaired in relation to the anchor. This could either be an absolute value (for example 4 hrs), or a mean or a median value of all the conforming relations from the event log between the anchor activity and the activity to be repaired according to the modeled labeled free-choice workflow net fragment. The conforming

Table 8.3: Repair actions for correcting ordering of activity e of the event log from Figure 8.4b using Figure 8.5.

Repair action	Anchor	Position	Value
1	b (t_1)	after	45 minutes
2	c (t_3)	before	45 minutes

relations are computed using alignments [5]. An absolute value is necessary in cases wherein the modeled fragment cannot be aligned with any of the traces containing the anchor activity and the activity to be repaired.

A repair action thus consists of an anchor activity from the modeled process fragment, the position, and the value. Multiple repair actions can be specified for repairing an activity for the same process fragment. The order of repair actions determines the sequence in which repairs are performed in the event log, such that if the first action fails, the second one is applied and so on. Table 8.3 shows an example of the repair actions for the example from Figure 8.5.

In the next section, we discuss the actual procedure followed to change the event log based on the configurations specified in this section.

8.5.3 Perform Repairs

The actual repair is performed using the outcome from the alignments based conformance technique [5]. An alignment is a sequence of moves. A move is a pair with an event or >> as first element and an activity or >> as second element. A projection of an alignment on its first elements (abstracting any >> away) yields the trace. A projection on an alignments second elements (abstracting any >> away) yields a visible execution path in the process model.

Aligning events belonging to a trace with a process model can result in three types of so called *moves* - synchronous move, move on model and move on log.

- **Synchronous move (S_m):** The occurrence of the next event in the trace is matched by an occurrence of a corresponding activity in the process model.
- **Move on model (M_m):** The occurrence of an activity in the process model is not matched by the next event in the trace. The move on model is *harmless* if the activity in the process model represents no behavior, i.e., it is a silent transition.
- **Move on log (M_l):** The occurrence of the next event in the trace is not matched by a corresponding activity in the process model.

Typically, *costs* are associated with M_m and L_m, while S_m and silent transitions have no associated costs. The cost of an alignment is thus the sum of all the costs of moves. An alignment is optimal if and only if the costs are minimal. Figure 8.7 shows the usage of alignments.

Table 8.4: The filtered event log from Figure 8.4 containing the activities b, c and e.

Case	Activity	TimeStamp
1	e	1-1-2018 08:00
1	b	1-1-2018 09:00
1	c	1-1-2018 11:00
2	c	2-1-2018 11:00
2	e	2-1-2018 12:00
3	b	4-1-2018 09:00
3	e	4-1-2018 10:10
3	e	4-1-2018 11:10
3	c	4-1-2018 12:00
4	b	5-1-2018 09:00
4	c	5-1-2018 12:00

Having provided an overview of alignments, we now demonstrate the use of alignments strategy to perform the actual repair of the event log with the help of the example shown in Figure 8.4b.

For each misaligned trace, all the repair actions are exhaustively applied, until the ordering is corrected, or there are no more repair actions remaining. The workflow followed for performing repairs is shown in Figure 8.6. For each repair action, the ordering of events could be changed in case of misaligned events, using the specified repair action (Change ordering in Figure 8.6). Moreover, in cases where there is extra behavior, some of the events could be removed (Remove events in Figure 8.6), or new events could be artificially introduced (Add events in Figure 8.6). The ordering information for the newly introduced event is based on the current repair action.

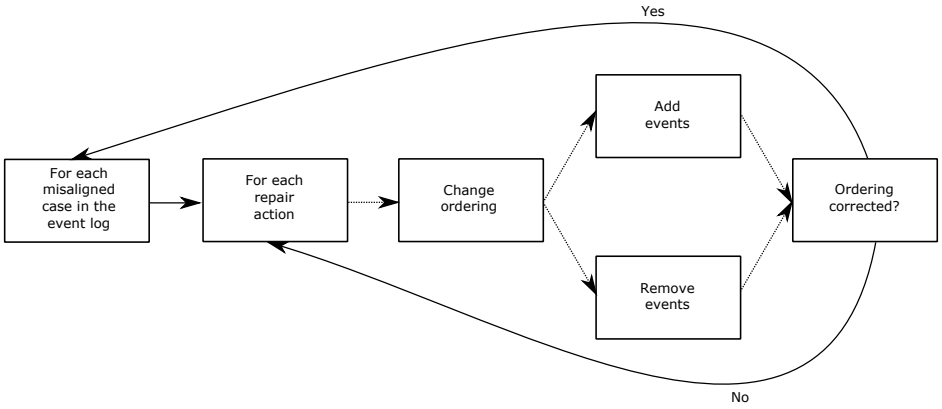


Figure 8.6: Approach for performing repairs of event ordering imperfections. Dotted arcs indicate optional steps.

Finally, the changed trace is re-evaluated against the labeled free-choice workflow net fragment. If the current repair action *fixes* the ordering within the trace, then we replace the original trace in the event log with the changed trace. By fixing the order, we mean that the trace and the model can be perfectly aligned, i.e., without having any log moves or (non-silent) model moves. If the repair action does not fix the ordering according to the specified labeled free-choice workflow net fragment, then we move on to the next repair action. This is repeated until all the repair actions are evaluated. If none of the repair actions fix the ordering, then the trace remains incorrectly ordered in the event log. This is repeated for all the traces of the event log.

Let us revisit our running example from Figure 8.4. Let the fragment from Figure 8.5 be the fragment modeled by the user, and let e be the activity set to be repaired. Furthermore, let us assume that the user allows for the addition of or removal of events as per the repair configuration. Let Table 8.3 denote the repair actions. Using this, the following steps are followed for each misaligned case in the event log:

Case # 1: We begin with the first repair action. In order to perform the *Change ordering* step from Figure 8.6, we begin by forming pairs between *move on model* of e and *move on log* of e, starting from the left-most side of the alignments of Figure 8.7, which results in a single pair of (step 1, step 3) for case # 1. For each pair, we find the closest synchronous occurrence of anchor activity corresponding to the *move on log* of e. The closest synchronous occurrence of the anchor activity is searched for on the left-side of *move on model* of e, if the position of repair action is set to *after*, and is on the right-side otherwise. In our case, the anchor happens to be step # 2. Now, we get the corresponding event from the trace in the event log that corresponds to the step 2. Next, we change the timestamp from the *move on model* element of the pair, by adding 45 minutes to the timestamp of the anchor event (b). Hence, the new timestamp of event e in case # 1 becomes “1-1-2018 09:45". Since there are no extra *move on model* or *move on log* of activity e, the steps *add events* and *remove events* are skipped. Upon aligning the changed trace and the labeled free-choice workflow net, we get a

step number	1	2	3	4
log sequence	e	b	>>	c
model sequence	>>	b	e	c

(a) Case # 1.

step number	1	2	3	4
log sequence	>>	>>	c	e
model sequence	b	e	c	>>

(b) Case # 2.

step number	1	2	3	4
log sequence	b	e	e	c
model sequence	b	e	e	c

(c) Case # 3.

step number	1	2	3
log sequence	b	>>	c
model sequence	b	e	c

(d) Case # 4.

Figure 8.7: The alignment outcomes of the cases in the event log shown in Table 8.4, corresponding to the model from Figure 8.5. For the sake of convenience, we do not show the move on models for the silent transitions ($t_4, \top$ and $\bot$) here.

Table 8.5: Event log after performing the repair actions.

Case	Activity	TimeStamp
1	b	1-1-2018 09:00
1	e	1-1-2018 09:45
1	d	1-1-2018 10:30
1	c	1-1-2018 11:00
2	a	2-1-2018 10:00
2	c	2-1-2018 11:00
2	e	2-1-2018 12:00
3	b	4-1-2018 09:00
3	d	4-1-2018 10:00
3	e	4-1-2018 10:10
3	e	4-1-2018 11:10
3	c	4-1-2018 12:00
4	b	5-1-2018 09:00
4	e	5-1-2018 09:45
4	d	5-1-2018 10:00
4	c	5-1-2018 12:00

perfect alignment, thereby correcting the ordering of events in case # 1. Hence, the next repair action is skipped, and we move to the next case.

Case # 2: Case # 2 does not have an occurrence of activity *b* at all. Since activity *b* is mandatory according to the fragment of Figure 8.5, no matter what changes are made to the timestamps of event *e*, the repaired trace would always be misaligned with the process fragment (due to the absence of *b*). Hence case # 2 will be uncorrected.

Case # 4: Since Case # 4 does not have the mandatory occurrence of event *e*, it could easily be introduced by using the first repair action and *adding* a new event. The value of the timestamp corresponding to the newly added event will be "5-1-2018 09:45". Hence, the repaired trace would be perfectly aligned in the process fragment of Figure 8.5, thus the next repair actions would be skipped.

The repaired event log is shown in Table 8.5. In the repaired event log, the value of the timestamps of the event *e* corresponding to Cases 1, 3 and 4 are correct. However, the value of the timestamp corresponding to Case # 2 is still incorrect. The user may thus decide to model a different fragment and/or specify different repair actions, in order to correct more such issues.

Having performed the changes to the event log, the user is then informed about the impact of the change, as discussed in the next section.

8.6 Impact Analysis

In this section, we discuss the impact of changes made to the event log, by comparing it to the prior event log (before the changes were made). Based on the impact of repair, the user can then choose to keep the changes in the event log, or revert back to the prior version of the event log (i.e., ignore the changes). The whole process of repair can now be repeated for the replaced event log and corresponding to the new problems detected. The metrics presented to the user to analyze the impact are as follows:

- Edit distance: Levenshtein distance between the repaired event log and the original event log.
- Fitness [5] of the original filtered event log and the repaired filtered event log for the activities from the labeled free-choice workflow net fragment.
- The total number of cases impacted, i.e., the total number of cases for which at least one repair action resulted in the correct ordering of events.
- The total number of events added.
- The total number of events removed.
- The total number of events with changed timestamp value.

8.7 User Interface and Evaluation

The proposed approach is also supported by the package in the nightly build version of ProM [171] in the *InteractiveProcessMining 6.8.56* package, under the plug-in "Data-quality". This plug-in requires an event log in XES [3] format as its input. This plug-in should ideally be used before performing any process mining analysis, in order to analyze and correct the possible issues in the event data. After correcting the possible data quality issues in the event log, we can then interactively discover process models using the approach discussed in Chapter 5–8.

Figure 8.8 shows the detection tab of our tool. The user is presented with all the detected quality issues (along with their description, frequencies etc.) in a tabular format (panel A). Upon selecting an issue from the detection table, the user is provided with the filtered event log (panel B) which contains only the events from the selected issue. Furthermore, the user is also provided with an aggregated histogram (panel C) showing the distribution of all the affected events based on the type of the detected issue.

Figure 8.9 shows the repair screen of our tool, which is the second tab of the same plug-in. The user models a labeled free-choice workflow net interactively by adding one activity at a time (panel A). This labeled free-choice workflow net specifies the relation between the activity to be repaired with other activities. The activity to be repaired is selected from a dropdown box by the user. The user specifies the

repair strategy in repair configuration view (panel B). Upon performing the repair, the impact is presented to the user (panel C) which can be used for the decision making of either keeping or ignoring the changes.

The approach presented in this chapter is evaluated using two publicly available real-life event logs. Detecting event ordering anomalies and fixing them essentially addresses the re-structuring of the event log to correct the control-flow aspect (of the underlying process model). Hence for each event log, we demonstrate the detectabi-

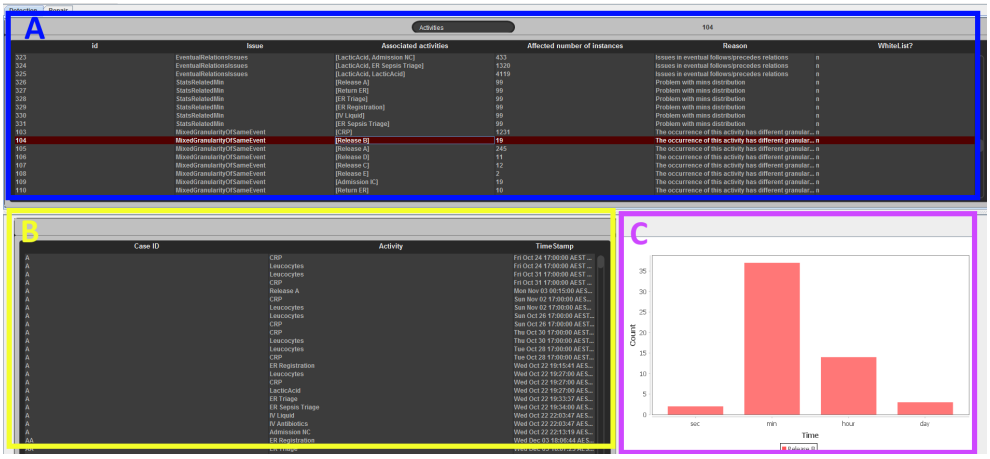


Figure 8.8: The detection tab showing the three detection panels: A) the list of detected issues, B) the event log view, and C) the graph view showing the distributions for the selected issue.

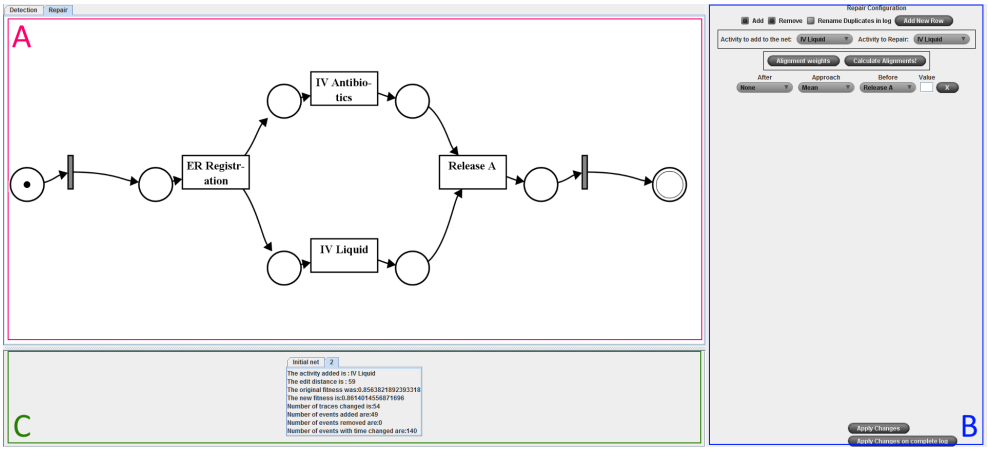


Figure 8.9: The repair tab showing the three repair panels: A) the Petri net modeled interactively by the user, B) the repair configuration view, and C) the impact of a (sequence of) repair(s).

lity of ordering related issues in the event log, followed by repairing the event logs using the insights from the detection phase along with the domain knowledge. The final outcome is evaluated based on the process models discovered using the repaired event log.

The event logs used are (i) the Sepsis event log [117]- wherein the process model discovered from a repaired event log is compared with the ground truth process model (available at [117]), (ii) the BPIC 2015 event log - wherein the outcomes of automated process discovery techniques are compared before and after the event log repair.

8.7.1 Sepsis

The Sepsis event log described in [117] contains the Sepsis treatment process of patients of a Dutch hospital. In total, the event log contains data of 1050 patients and 16 activities.

The authors from [117] designed a process model *by hand* with the guidance of domain experts which resulted in a process model as shown in Figure 8.10. However, as noted by the authors in [117], in order to come up with a final process model they had to undergo multiple iterations to find a balance between considering the reality depicted by the data and the protocols according to the domain expert. At each iteration, the authors manually modeled a process model using insights from the domain experts, and performed alignments to understand the correctness of the process model. That is, the authors tried to manually find the best fit between the data and the domain expert, by modeling an end-to-end process model at each iteration. However, the comprehensibility of a process model reduces upon increasing the size of the process model. Hence, evaluating the complete end-to-end process model in each iteration with expensive resources such as doctors in a hospital is both a time consuming and a costly affair.

In this chapter, we first identify the plausible issues in the event log, fix them with the help of domain knowledge and hand over the actual discovery task to discovery

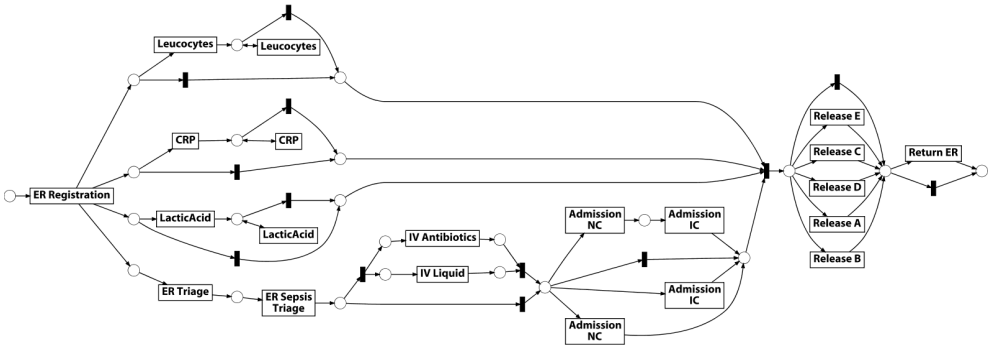
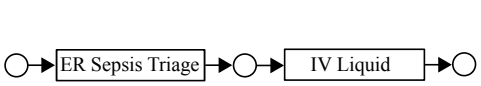
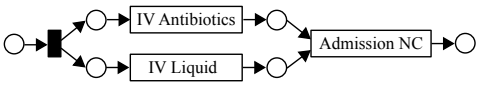


Figure 8.10: Original Sepsis model for the event log as modeled in [117].



(a) Relationship between the activities *ER Sepsis Triage* and *IV Liquid*.



(b) Relationship between the activities *IV Antibiotics, IV Liquid* and *Admissions NC*.

Figure 8.11: Process fragments modeled to repair the Sepsis event log.

algorithms. In our case, the users model their domain knowledge through process fragments instead of analyzing the complete end-to-end process model.

In total, the timestamp ordering of six activities was corrected. During the correction phase, mostly the default settings were used. The only exception was to allow the possibility of removing extra events, which is not the case by default.

Here, we give an example of the steps followed to repair one of the activities. The detection outcome of our tool indicated that there were quite a lot of ordering related issues in the original event log. Among them, almost 50% (14 out of 28) of the ordering problems had at least one of the three activities: *IV Antibiotics, IV Liquid*

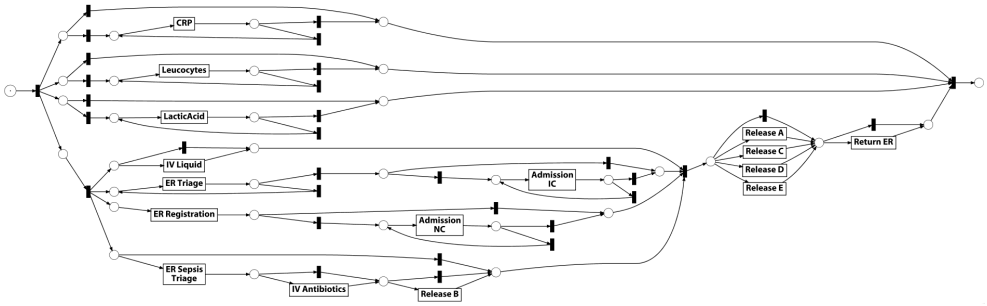


Figure 8.12: Sepsis model discovered from the original event log using the inductive miner-incomplete discovery algorithm.

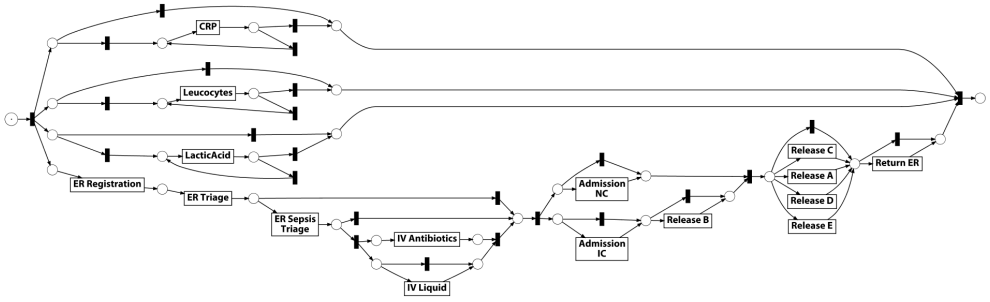


Figure 8.13: Sepsis model discovered from the repaired event log using the inductive miner-incomplete discovery algorithm.

Table 8.6: The steps followed in order to repair the Sepsis event log, and its impact compared to the original event log.

Activity Repaired	Edit distance	% traces impacted
ER Triage	15	0.9
ER Sepsis Triage	46	2.3
IV Liquid	124	6.2
IV Antibiotics	22	1.1
Admission NC	407	32.3
Admission IC	17	1.2

and/or *Admissions NC*, which hinted at a possible ordering problem with regards to these activities.

There are certain protocols which must be followed in a Sepsis treatment process. We used the process models from [117] to identify a few such protocols that must be followed in a Sepsis treatment process, which served as our 'domain knowledge' for this evaluation. These protocols were then translated into Petri net fragments as shown in Figure 8.11. Figure 8.11 shows two of the fragments designed in order to repair the activities *Admissions NC* and *IV Liquid*, based on the insights gathered. Figure 8.11a was used to correct the timestamps of *IV Liquid* in the event log, in relation to *ER Sepsis Triage*. A single repair action was used to configure *IV Liquid* to be after (the mean duration of all the fitting cases) *ER Sepsis Triage*. A similar fragment and repair action was used, by replacing *IV Liquid* with *IV Antibiotics*, in order to correct the timestamps of *IV Antibiotics* with respect to *ER Sepsis Triage*. A similar approach was used to correct the ordering of *ER Sepsis Triage* to be after *ER Triage*, and *ER Triage* to be after *ER Registration*. In order to repair the activity *Admissions NC*, the position of *Admissions NC* was configured to be after (the mean duration of all the fitting cases) *IV Antibiotics* or *IV Liquids* for the misaligned cases, by using two repair actions, one corresponding to *IV Antibiotics* and another one corresponding to *IV Liquid* as the anchor event. A similar process fragment was obtained by replacing *Admission NC* with *Admission IC*, and similar repair actions were applied to repair the incorrect occurrences of the activity *Admission IC* in the event log.

Table 8.6 shows the impact of performing the changes. Almost 30% of the cases were impacted by the activity *Admission NC*. This is also in keeping with the alignments which were performed using the original event log and the original process model from Figure 8.10, which resulted in about 30% of the cases having either move on log with respect to the activity *Admissions NC*.

After repairing the time values (and removing the duplicates) of the 6 activities, the final repaired event log was used to automatically discover a process model. Given the incomplete nature of the log we chose the Inductive miner incomplete [110] discovery algorithm to discover a model which was closest to the original process model. The result is shown in Figure 8.13.

It should be noted that using the original event log (before repair) to discover a

process model using inductive miner incomplete resulted in a process model wherein almost all the activities could occur multiple times (self-loops) and in parallel with other activities as shown in Figure 8.12. Upon comparing the process models in Figure 8.10 and 8.13, it is quite evident that by using our tool to repair the event log in just 6 steps, we were able to come very close to the original process model as discovered manually by the authors of [117].

It could be argued that similar results could be obtained by using conformance checking techniques, and correcting the misaligned events, such as the one proposed in [140]. Such techniques could of course be used in order to automatically repair an event log when a complete end-to-end process model is available. However, the fundamental difference is that we assume that we do not know the complete end-to-end model beforehand. Instead, we merely use some of the fragments in order to improve the quality of the event log, and ultimately improve the quality of the complete end-to-end model discovered by automated process discovery techniques.

8.7.2 BPIC 2015

The second event log that we use to evaluate our approach is from the BPI Challenge 2015 [65] which contains five event logs corresponding to the building permit applications of five municipalities in the Netherlands. We use the event log from one of the municipalities (municipality 1) which contains almost 400 activities. In order to make the results comprehensible, we filtered the log and selected the top-22 frequent activities belonging to the main application process. Also, we focus only on those cases which are *closed*, which results in a total of almost 700 cases.

Upon loading the filtered event log in our tool, the detection technique hinted at a possible problem in the granularity of events in the event log: with more than 10 activities having multi-granular timestamp values across the event log as shown in Figure 8.14. It is quite noticeable that several activities have many events registered at *hourly* granularity. This is very suspicious and thus resulted in the loss of ordering

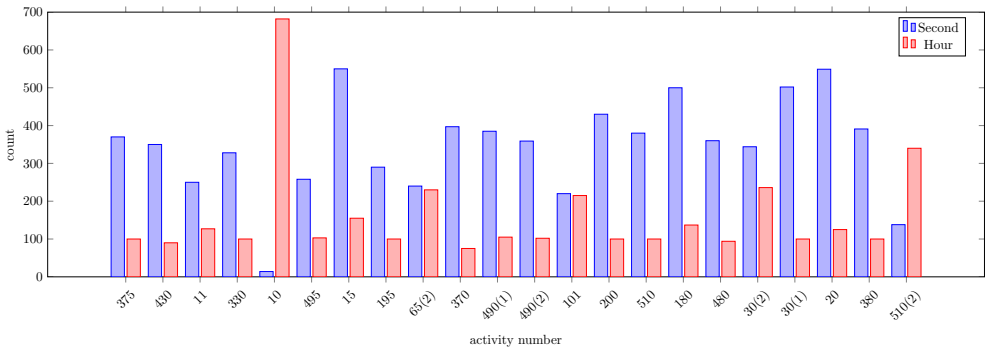


Figure 8.14: The distribution of the number of events of an activity having hourly granularity vs second level granularity.

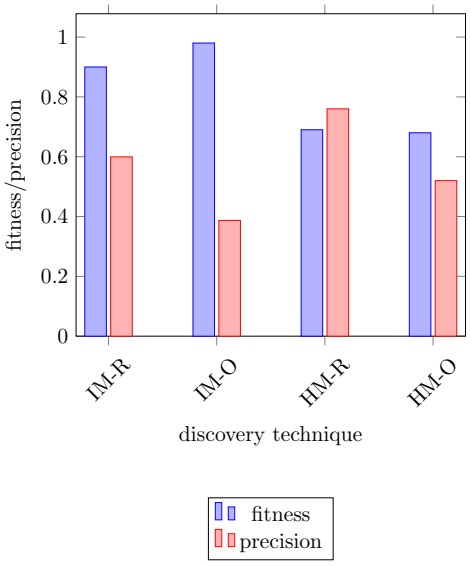
information in the event log. Therefore, we try to repair the event log in order to manually correct the ordering of activities.

All the activities in the event log have the following naming structure: 01_HOOFD_###. Some activities also have an additional _## at the end. We know that typically the last three digits from 01_HOOFD_### denote the ordering sequence of activities (however this may not always be the case). We use this information as our *domain knowledge* in order to *repair* the positioning of activities which have high number of events with hourly granularity, in relation to the activities which have high number of events with seconds granularity. Upon doing this repeatedly, we increased the granularity of almost 45% of the events which initially had hourly granularity.

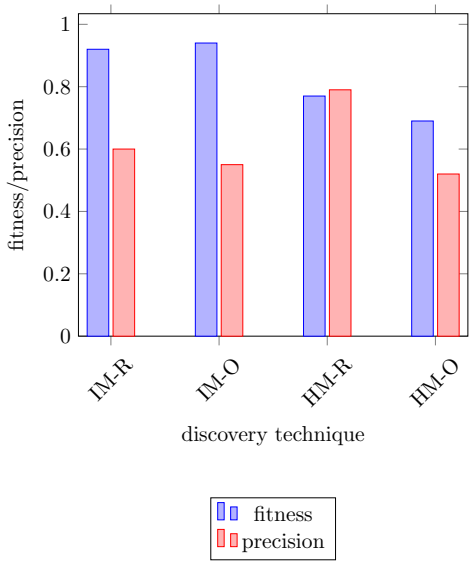
Contrary to the Sepsis event log, in the case of BPIC 2015, we do not have a *ground truth* process model for comparison. Hence, we use the process mining quality dimensions of fitness and precision to evaluate the results.

We discovered process models using two state-of-the-art process discovery algorithms (the Heuristic miner [185] and the inductive miner [110]) using the original (non-repaired) event log at default settings. Next, we used the same algorithms to discover process models using the repaired event log.

All four discovered process models are evaluated against the original (non-repaired) and repaired event logs in order to assess them based on fitness [5] and preci-



(a) The fitness and precision values evaluated against the original event log.



(b) The fitness and precision values evaluated against the repaired event log.

Figure 8.15: Comparing fitness and precision scores. IM and HM denote the Inductive miner infrequent and the Heuristic miner discovery algorithms resp. The leading -R and -O denote the event logs (repaired and original resp.) used for discovery.

sion [128] dimensions. The results are shown in Figure 8.15. It should be noted that, there is typically a trade-off between fitness and precision of a process model. In Figure 8.15b, we compare the process models against the repaired event log. Unsurprisingly, we see that the process models discovered using repaired event logs perform better than the process models discovered using the original event logs. However, in Figure 8.15a when the process models are compared against the original event log, we see that the process models discovered using the repaired event log have a much higher precision even when evaluated against the original event log, as compared to the process models which were discovered using the original event log. Furthermore, there is a negligible difference in the fitness scores. This indicates that the process models discovered using the repaired event log are of much higher quality, irrespective of the event logs that they are evaluated against. This is significant because the user may decide to discover an accurate process model based on the repaired event log. However, may decide to perform further analysis, for example, conformance/performance analysis, based on the original event log.

Similar results could possibly be obtained by using some automated filtering-based techniques from the literature, such as [45], which remove *noisy* behavior from the event logs. However, in our approach, the user is central to decision-making. This is especially important as the user can use the domain knowledge in order to correct the timestamp ordering of events, and not rely on some black-box technique to make decisions. Moreover, by involving the user directly in the process of event log repair, we allow the user to have more *control* over the overall repair process. That is, in automated techniques, the user has very little intuition of what was changed in the event log. However, in our approach, the user is involved in each step of repairing the event log, and is provided with the feedback about the impact of changes made in the event log.

In the next section, we discuss pre-existing approaches described in the literature which are relevant for this chapter.

8.8 Related Work

The manifestation of data quality problems in event logs in the context of process mining has been well researched in literature, and is summarized in Section 8.2. [21] was one of the first approaches which characterizes data quality issues in the event log. This was later extended in [113] and [163]. [113] also introduces the quality issues in a healthcare setting. In [151], the authors explore and manifest the typical event data quality issues. Our work does not propose new data quality issues, but instead focuses on detecting and interactively repairing the time related event log quality issues. Hence, in this section, we compare our technique with the techniques from the literature which allow (i) detection and/or (ii) repair of event order related data quality issues in the event log.

8.8.1 Detection of Quality Issues

Detection of possible quality issues in the event log, is a well-researched topic, both within the process mining community, and outside it.

Process Mining

Many process mining techniques implicitly detect quality issues in the event log. For example, conformance checking technique discussed in [5] matches expectation as modeled by a Petri net with reality represented by an event log, to detect conforming and deviating behavior. Thus the order related data quality issues would surface as deviating behavior. However such techniques *require* a process model (either procedural or declarative) to begin the analysis with. Our approach on the other hand does not require ground truth during the detection phase.

Many process discovery algorithms (for example, [95, 110, 185]) implicitly try to detect noise in the event log, which can sometimes be attributed to event ordering imperfections. However, the decisions and detections made during the discovery phase are implicitly incorporated in the discovered process model, but not explicitly presented to the user. Our approach explicitly presents the user with the individual anomalies related to event ordering imperfections.

Other Approaches

Techniques such as [193], allow users to visually explore sequences of events by providing an overview of the events using various visualization techniques [70], such as state diagrams, icicle plots etc. Some of these techniques particularly focus on identifying anomalies such as anomalous events in computer networks [27, 28, 133] or in healthcare domains [17]. The user can perform interactive operations such as, search, filter, group sequences etc. in order to visually explore event sequences and patterns [89, 109]. The insights gained from such interactive explorations could then be combined with the domain knowledge in order to visually identify possible ordering-related data quality issues in the event log. However, these techniques typically show what has happened in the event log, and the onus is upon the user to identify any possible data quality issues in the event log. Some techniques from the literature automatically quantify the quality of an event log. [153] is a package in R which provides aggregated information about the structuredness and behavioralness of an event log. [20] provides a technique for interactively assessing data quality, and using customizable metrics to quantify the quality issues. Similarly [103] discusses various metrics to quantify the overall quality of the event log and [48] suggests several data quality metrics and data cleaning strategies. However, these techniques typically provide a global measure of data quality and do not pin-point the exact list of issues within the event log. Our approach focuses on providing the user with a list of time-related quality issues in the event log.

8.8.2 Event Log Repair

Unlike process discovery and performance analysis, repair of data quality issues in event logs is fairly unexplored territory in the domain of process mining. The authors of [140, 141] describe techniques to correct the positioning of events based on (timed) Petri nets and probabilities derived from alignments information. Similarly, in [150] the authors describe a Petri net decomposition-based heuristic repair strategy for efficient event log repair. However, in reality, a de-facto standard process model is seldom available. Compared to these approaches which assume the presence of a process model, our approach does not require an end-to-end process model for event log repair.

In [41] and [149], the authors describe a denial-constraint-based approach and a temporal-constraint-based approach to automatically repair the timestamps of events in the event log. Again, these approaches require the users to pre-specify the constraints, whereas in our case the user can first analyze the detected issues, and then choose to act upon them.

In [92] a visual-guided approach for repairing time related issues in event log is discussed. Contrary to both constraint-based approaches and visualizations driven approaches to event log repair, our approach allows the user to flexibly specify the domain knowledge using graphical process fragments, which typically are more intuitive and allow specification of complex process behavior such as concurrency, loops, choices, duplication of activities etc. It could be an idea to combine the repair approach proposed by us, with the constraint-based repair approach and the visualization-driven repair approach, to support broader repair strategies.

In [45, 79, 80], techniques are proposed to automatically remove/fix the “noisy" behavior from the event log without any user involvement. However, in our case we present such probable issues to the user, instead of automatically removing them from the event log. The user can temporarily repair the issues, analyze the impact and optionally make the changes permanent. Thus, our system provides an integrated detection and repair mechanism that allows the user to interactively and incrementally repair event logs.

8.9 Conclusion and Future Work

In this chapter, we presented an integrated technique for first detecting imperfections in event ordering in the event log, followed by an interactive repair strategy to correct the ordering. Based on a literature review, the symptoms of event ordering imperfections were discussed followed by a comprehensive strategy to detect such symptoms. The proposed repair approach allows the user to develop a global truth (designed as a process fragment), which can be enforced locally on each case in the event log. This makes our repair approach easier compared to a case-by-case repair strategy. Furthermore, intuitive graphical process models can be interactively designed based on the strategies discussed in Chapter 5–Chapter 9, for easy incorporation of domain knowledge, without the need of working with a non-trivial end-to-end

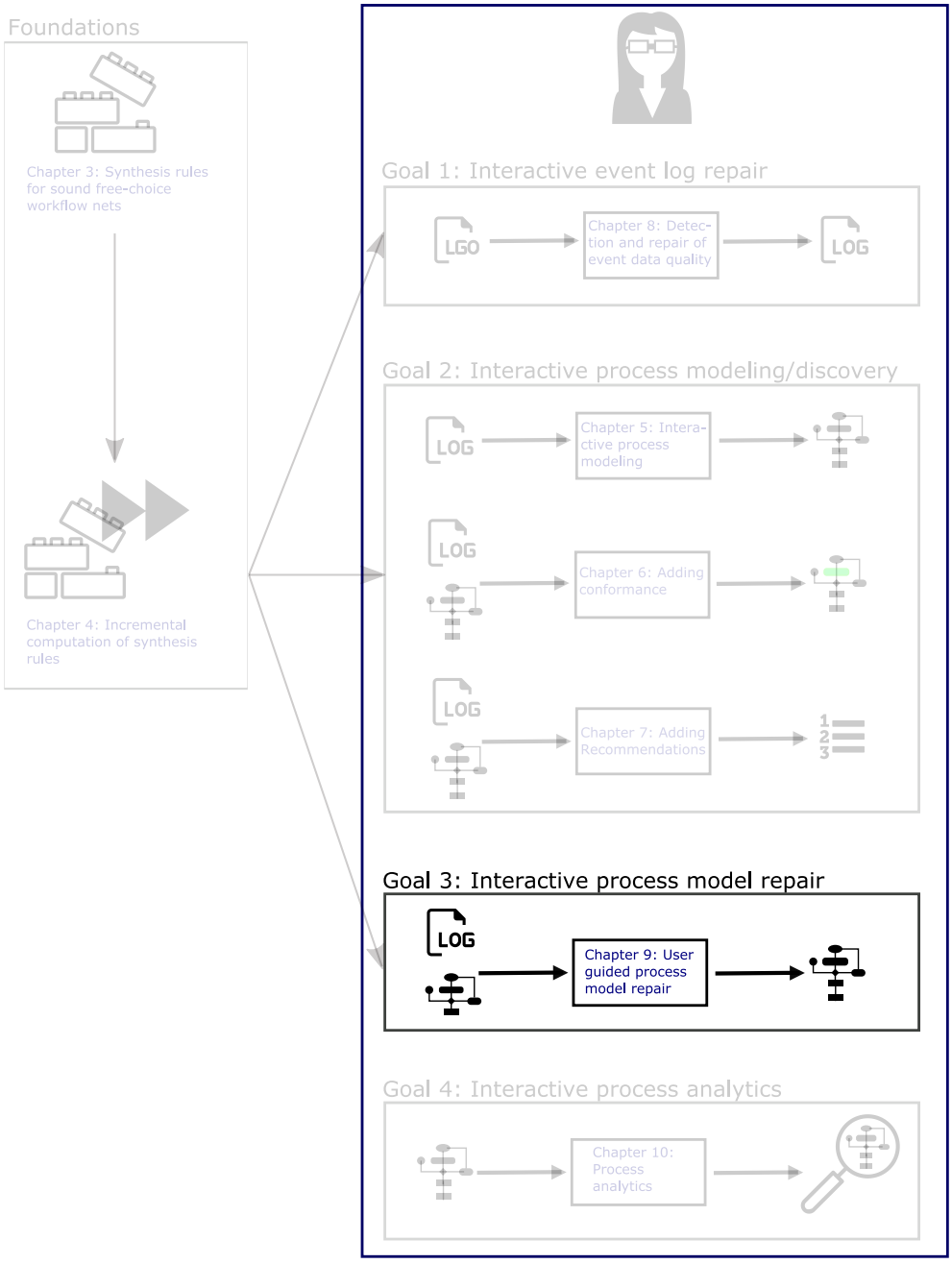
process model.

We applied our detection and repair techniques on two real-life event logs, and showed that we can detect anomalies in the event log related to event ordering imperfections, and re-structure the event log to repair these anomalies. Process discovery techniques benefit greatly by fixing the data quality issues in the event logs, as we were able to repair the *source* of the problem.

Finally, we would like to propose some future directions. In the detection phase, the user is presented with a host of detected issues. However, it might be the case that the number of detected issues is too high. Even though we allow the user to sort and search through the detected issues, in the future, it would be ideal to smartly filter some of the irrelevant or over-lapping issues before presenting them to the user. Another future direction could be to allow the user to semi-automatically repair the event log. In the current version, we depend entirely on the user to repair the event log. However, in cases when an end-to-end process model is available, it may be ideal to use the outcome of conformance analysis and semi-automatically repair the incorrectly ordered events. That is, the user may want to use the insights from conformance analysis, in combination with the domain knowledge, in order to decide which events should be repaired based on the control-flow of the end-to-end process model. Another interesting future direction could be to dive deeper into the performance and compliance perspective. Currently, we only look at the control-flow aspect of an event log. An event log may additionally contain many more event-level and case-level attributes. Hence, in the future, it would also be ideal to interactively explore and repair other data quality (non-control flow) issues in process mining. Finally, it would also be desirable to provide indicators about the root cause, i.e., smart detection mechanisms to further assist the user in exploring what might cause possible issues in the event log.

Chapter 9

User-guided Process Model Repair



This chapter is based on the publications [60] and [61].

In this chapter, we discuss strategies to repair a pre-existing process model. The pre-existing process model could either be drawn *by hand* by interviewing different actors involved in the different stages of the process. Alternatively, the process model could be discovered automatically using information from the event log with an automated process discovery technique. The strategy used for repairing the process models could be based on the way in which the process model was constructed in first place.

On one end, consider a process model which is drawn by hand, using the traditional approach of process model construction with interviews. When the event data is recorded by the information systems, the hand-drawn process model can be compared with the event logs to firstly detect the points in the process where the process model might deviate. Next, the process model could be *changed* in order to repair the observed deviations.

On the other end, consider a process model which is discovered using an automated process discovery technique. In such a case, the reality as depicted by event logs is already taken into account by the discovery techniques. However, as discussed previously in Chapter 8, there may be unsolved data quality issues in the event log which may lead to discovery of incorrect process models. Furthermore, the representational biases of discovery techniques may limit the types of process models discoverable. Moreover, the discovered process model may be too difficult to understand for the end user. Hence, in such cases, the user should be able to *repair* the process model as per the user's domain knowledge.

The focus of this chapter is to discuss strategies to repair pre-existing process models using information from the event logs and domain knowledge. The remainder of the chapter is structured as follows. In Section 9.1, we discuss the main problem addressed in this chapter. In Section 9.2, and Section 9.3, we provide approaches to address the problems discussed in Section 9.1. In Section 9.4 we discuss the implementation details and evaluate the proposed approach using a synthetic and real-life event log. In Section 9.5, we discuss some related work, followed by conclusions in Section 9.6.

9.1 Problem Definition: Process Model Repair based on Domain Knowledge

Goal 1 from Chapter 1 dealt with enabling interactive repair of event logs and Goal 2 from Chapter 1 dealt with interactive process modeling. However, there may be cases wherein a process model already exists, and such a process model needs to be repaired based on the insights from the real-executions recorded in the event log, or purely based on the domain knowledge. The problem of process model repair is not new, and has been well researched in the literature. However, most of the techniques focus on correcting the process models solely based on the information from the event logs. In our case, we would also like the user to have a larger influence in repairing the process models. This chapter aims to address Goal 3 from Chapter 1, to enable interactive repair of preexisting process models using domain knowledge along with

the information from the event logs.

- How can we enable interactive process model repair, such that:
  - domain knowledge can be implicitly added to repair a process model. Typically, in such scenarios, a process model is discovered automatically using some automated process discovery technique. Hence, the user has very little influence over the actual discovery of the process model. However, after the discovery, the user may want to edit such a process model in order to *correct* some of the possible issues. Hence, it is vital that the technique supports editing of preexisting process models. Moreover, all the requirements introduced in interactive process modeling scenario, such as broad structural representations, soundness etc., also hold true in the interactive process model repair scenario.
  - domain knowledge can be explicitly added to repair a process model. In such cases, a process model may already exist, but the user may want to ensure that it can also support the insights gained from the event log. However, relying completely on the event logs in order to repair process models, could lead to unanticipated and drastic changes in the process models. Hence, the user may want to ensure that certain constraints/rules are taken into account, when the process model is being repaired based on the information from the event log. Therefore it is vital that the technique proposed can find a balance between the provided domain knowledge and the information from the event logs in order to repair a process model.

In Section 9.2, we focus on the first aspect while using domain knowledge implicitly to interactively repair a process model. In Section 9.3, we focus on the second aspect, based on explicitly specified domain knowledge user specifies domain knowledge in terms of constraints, which are then used in combination with the event log to automatically generate repaired process models.

9.2 Interactively Repairing Process Models

In this section, we discuss an approach for interactively repairing pre-existing process models. It should be noted that the process models could be drawn by hand, or discovered automatically using any of the process discovery techniques. Given the advantages of free-choice workflow nets and synthesis rules, such as broad structural representations and guaranteed soundness, we use sound free-choice workflow nets in this section too.

In order to enable interactive editing of process models, we use the aspects of synthesis engine, labeled free-choice workflow net editing and conformance checking as described in previous chapters. The way in which the items from Chapter 5 and Chapter 6 are utilized is quite straight-forward and is shown in Figure 9.1.

The details of performing interactive repair, as shown in Figure 9.1 are as follows:

1. A process model, represented by a sound labeled free-choice workflow net is reduced into an atomic net by using reduction rules. Reduction rules, as discussed in Chapter 3, are the reverse of synthesis rules. Hence, a sound labeled free-choice workflow net can always be reduced to an atomic net, by using the reduction rules iteratively.

2. At each reduction iteration, track the changes made, using the so-called *reduction tracker*. That is, the type of reduction rule used, the label of the transition which is reduced (in case a transition is reduced), and the sequence number of reduction rule.

3. After reducing the sound labeled free-choice workflow net to the initial net, use the synthesis rules to synthesize the sound labeled free-choice workflow net using the synthesis engine. The sequence, type of rules, and activity labeling is extracted from the reduction tracker of the above step. The sequence of synthesis rule applications followed is the reverse of the reduction procedure of the above step. That is, starting with the atomic net, we now use the reduction rules in reverse (i.e., the synthesis rules) as tracked by the reduction tracker in reverse, in order to get the original labeled free-choice workflow net that we started with. The primary aim of steps 1-3 is to make the labeled free-choice workflow net editable in our synthesis engine which was proposed in Chapter 4.

4. Perform conformance analysis on the synthesized net. The results from conformance analysis indicate possible issues in the process model.

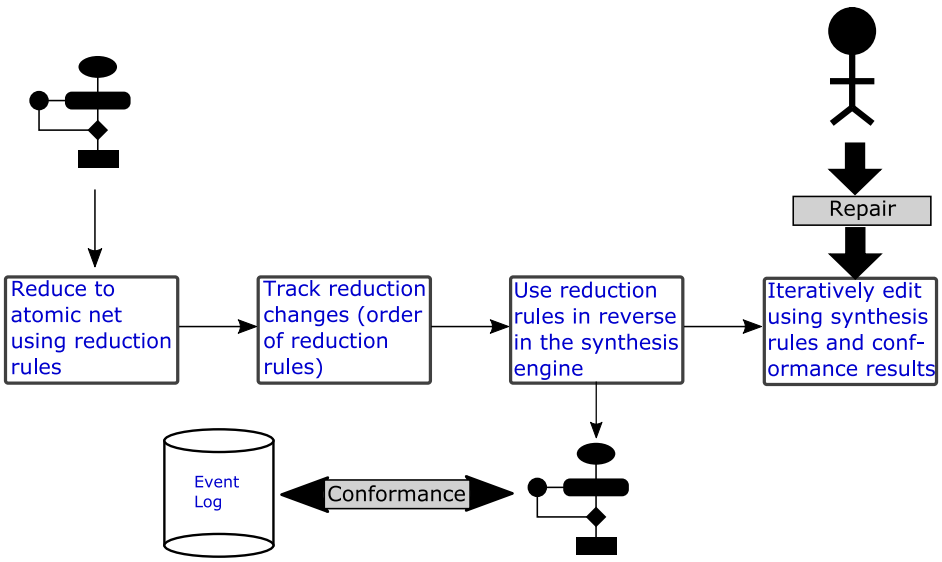
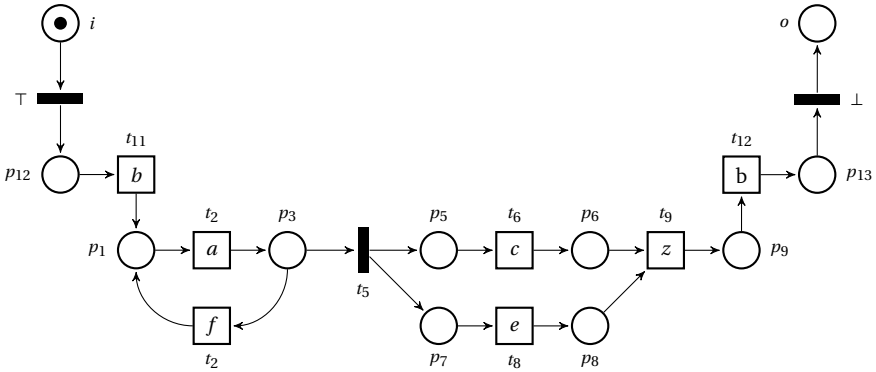
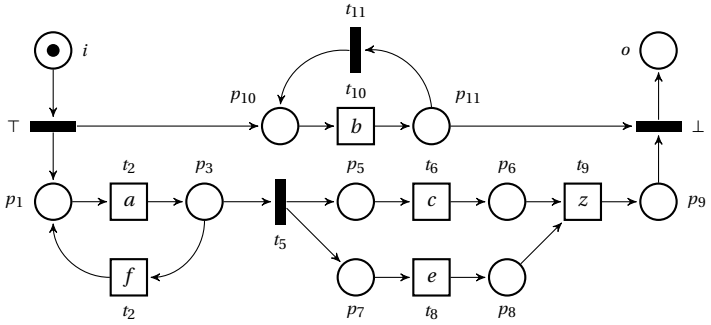


Figure 9.1: Interactive repair enabled by sound labeled free-choice workflow net editing based on synthesis rule engine.



(a) Expected process model in which activity *b* is repeated twice.

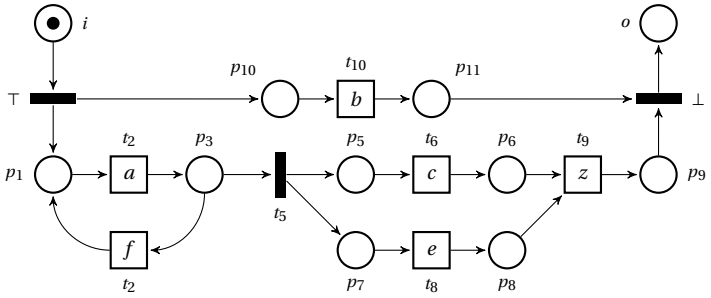


(b) Process model discovered using an automated process discovery algorithms. Many automated process discovery techniques cannot discover duplicate occurrences of activities.

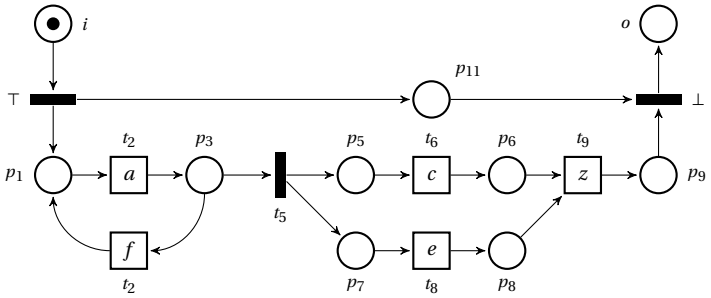
Figure 9.2: Expected and automatically discovered process models.

5. A user analyzes the issues in the sound labeled free-choice workflow net based on conformance analysis. The user can then iteratively edit the labeled free-choice workflow net by using synthesis rules as shown in Chapter 5 and conformance results as shown in Chapter 6 in order to interactively repair the process model. Furthermore, the user may remove parts of the net, based on the reduction rules (i.e., the reverse of synthesis rules). Upon using a reduction rule, steps 1–4 need to be recomputed, by populating a new reduction tracker. When a synthesis rule is used to expand the net, there is no need to recompute steps 1–4. We explain the approach presented in this section with an example.

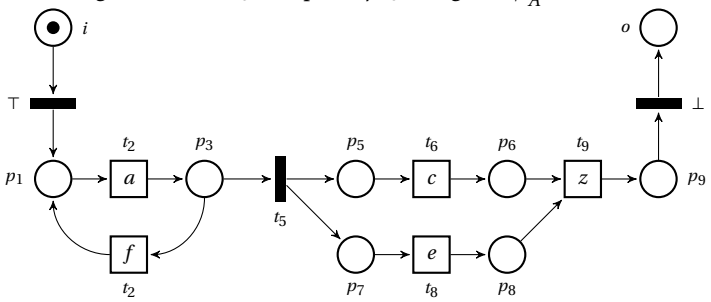
Let us assume that the labeled free-choice workflow net shown in Figure 9.2a is the expected labeled free-choice workflow net. Let us assume that some discovery algorithm discovered the labeled free-choice workflow net shown in Figure 9.2b based on the information from the event log. Clearly in the automatically discovered process model, activity *b* can happen more than two times, whereas in the expected process



(a) Reducing the net from Figure 9.2b by removing the transition t_{11} by using ψ_T^{WF} in reverse.



(b) Reducing transition t_{10} and place p_{10} using the ψ_A^{WF} rule in reverse.



(c) Reducing the place p_{11} by using the ψ_P^{WF} rule in reverse.

Figure 9.3: Reduction steps involved in repairing an automatically discovered labeled free-choice workflow net.

model, this activity happens exactly twice.

In order to repair the automatically discovered process model, to represent the expected process model, the user first has to *remove* the incorrectly placed loop from the process model. In order to do this, the user can use the synthesis rules in reverse, i.e., the reduction rules. Figure 9.3 shows these reduction steps. After using ψ_T^{WF}, ψ_A^{WF} and ψ_P^{WF} rule in reverse, we end up with the process model as shown in Figure 9.3c. The user can then use two applications of the ψ_A^{WF} rule (between $\{\top\}$ and

$\{p_1\}$ and $\{t_9\}$ and $\{p_9\}$), in order to add two instances of activity b in the labeled free-choice workflow net, in order to repair and model the expected labeled free-choice workflow net.

In the next section, we discuss an approach to automatically repair process models using domain knowledge.

9.3 Automated Process Model Repair based on Constraints

The interactive process model repair approach discussed in the previous section allows the user to interactively repair process models using the insights from the event logs, while incorporating the domain knowledge implicitly during process model repair. However, in certain scenarios, it would be ideal for the user to specify domain knowledge explicitly as some business rules, which could then be used to automatically repair a process model. It is necessary that the user can specify certain business rules that should hold in the process model, as letting an automated technique make the decisions entirely based on the event log could result in process models which are perfect based on the event log, however, may not satisfy any of the business rules.

An automatically repaired process model may satisfy all the constraints and may be correct according to the information from the event log. However, the user may find the repaired model to be too complex, or uninterpretable, as the comprehensibility of a process model is subjective in nature. Hence, it is vital that the user can choose a process model from multiple such repaired process models, which satisfy the constraints and the information from the event log to varying degrees.

Repairing process models in an exhaustive way would be too inefficient, and hence, not feasible in a practical scenario. Hence, we propose an approach to genetically create candidate process models. An overview of genetic approach is presented in Figure 9.4. A *population* of process models are created and evaluated against the

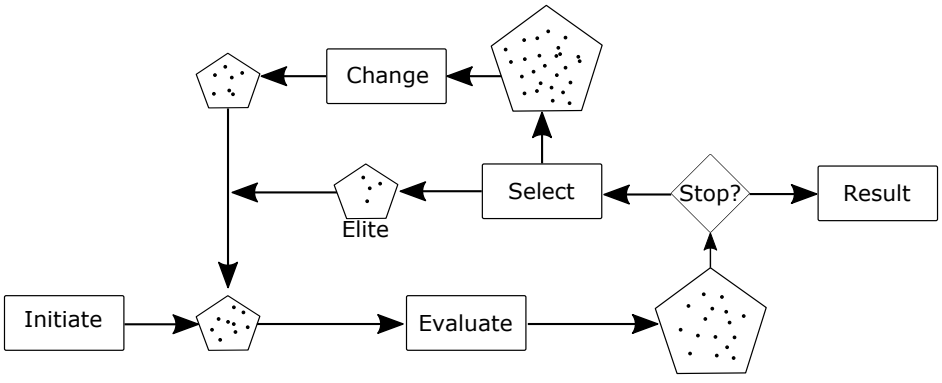


Figure 9.4: Different phases of genetic algorithm [24].

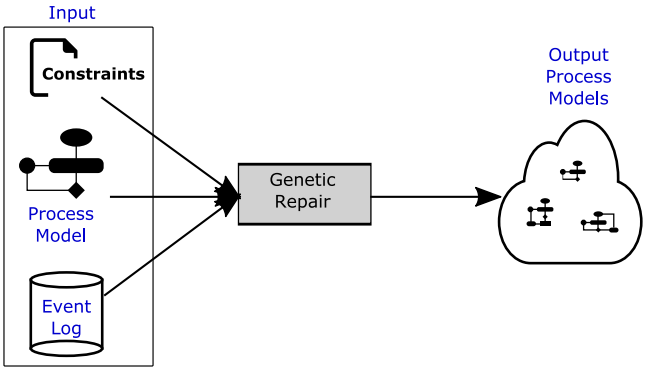


Figure 9.5: Input and outputs of genetic algorithm based repair approach.

quality dimensions compared to the event log based on conformance analysis, as well as based on the number of constraints (business rules) satisfied. The models from the populations are randomly changed using the so-called genetic operators. These models are re-evaluated until the stopping criteria is met. Finally, the user is provided with multiple process models, each having different qualities, based on the conformance scores and the number of constraints satisfied.

The input and output of the proposed approach are as shown in Figure 9.5. An event log, a process model and a set of constraints serve as the input to the technique. After the genetic approach terminates, the user is provided with a Pareto front of dominating process models.

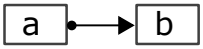
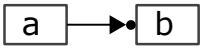
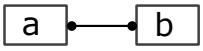
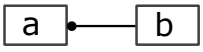
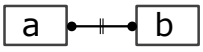
In the upcoming sections we provide details of different components from Figure 9.5. We start by discussing the use of Declare constraints, which provides a way to specify domain knowledge in our approach. We then discuss the process modeling notation used in our approach, and provide motivation for the same. We then briefly discuss the genetic approach used as well as the outcome of the approach, i.e., the Pareto front.

9.3.1 Declare Constraints to Incorporate Domain Knowledge

In this section, we discuss a way of providing domain knowledge. Sometimes, the user may not be aware of the actual control flow between activities, and hence cannot directly incorporate domain knowledge to interactively repair process models. In such cases, the user may be aware of certain business rules (or constraints) that should hold in the process model. Declarative constraints provide a handy way for modeling of such constraints. A declarative model is defined by using constraints specified by a set of templates [134]. We use a subset of *Declare* templates as a way to input domain knowledge.

Table 9.1 provides an overview and interpretation of the Declare constraints that we consider [116, 134]. Binary templates provide ways to specify dependency (positive and negative) between two activities. For example, $response(a, b)$ specifies that

Table 9.1: Declare constraints and their graphical and textual interpretations

Template Name	Graphical Representation	Interpretation
response(*a*, *b*)		Activity *b* should (always) eventually occur after activity *a*
precedence(*a*, *b*)		Activity *b* can (eventually) occur only after the occurrence of activity *a*
chain – *response*(*a*, *b*)		Activity *b* should (always) immediately occur after activity *a*
chain – *precedence*(*a*, *b*)		Activity *b* should (always) immediately be preceded by activity *a*
coexistence(*a*, *b*)		Activity *a* implies the presence of activity *b* (and vice versa)
responded – *existence*(*a*, *b*)		Activity *a* implies the presence of activity *b*
not – *coexistence*(*a*, *b*)		Activity *a* implies the absence of activity *b* (and vice versa)
not – *succession*(*a*, *b*)		Activity *a* should never be followed by activity *b*

activity *a* has to be eventually followed by activity *b* somewhere in the process. Six of the primary binary constraints considered are shown in Table 9.1. This subset of declare templates have been chosen to cover both positive and negative dependencies between activities. This set could later be extended using other declare templates as desirable.

9.3.2 Process Modeling Notation for Genetic Setting

Clearly, genetic algorithms are the ideal choice for repairing process models based on both the event log information as well as the number of constraints satisfied. The key step of genetic algorithms is to *modify* (Change step in Figure 9.4) certain aspects of existing process models in order to derive new candidate process models. It is

vital that the steps involved in the modification stage are not too time consuming. Furthermore, since we are interested in exploring process model space, it is ideal to change large parts of a process model at once.

The modification should also guarantee that only sound process models are created (the soundness subgoal of Goal 2 from Chapter 1). One way could be to use free-choice workflow nets, along with the synthesis rules, in order to ensure that only sound process models are generated. In Chapter 4, we have optimized the way in which the synthesis space is computed for a free-choice workflow net, based on the incremental synthesis structure. The incremental synthesis structure works well when we are interactively exploring a single process model. However, in a genetic setting, we are interested in exploring a lot of process models. Thereby, each such candidate process model would have its own incremental synthesis structure. This could lead to inefficiencies especially when the number of process models grows.

It is not ideal to randomly modify free-choice workflow nets (i.e., not based on synthesis rules), as it could lead to the construction of unsound process models. In order to overcome the drawbacks of using free-choice workflow nets in combination with synthesis rules in a genetic approach, we propose using process trees as the modeling notation, as process trees are sound by construction. We first introduce process trees, followed by the advantages of using process trees in a genetic setting, and finally discuss some of the limitations.

Process trees provide a way to represent process models in a hierarchically structured way containing operators (*parent nodes*) and activities (*leaf nodes*). The operator nodes specify the control-flow constructs in the process tree. Figure 9.6a shows an example process tree, corresponding to the labeled free-choice workflow net of Figure 9.2b. A process tree is traversed from left to right and top to bottom.

The order of child nodes is not important for the *and* ($\wedge$), and *exclusive-or* ($\times$) operators, unlike the *sequence* ($\rightarrow$) and *Xor-loop* ($\circlearrowleft$), where the order is significant. In the process tree from Figure 9.6a, the child nodes $\circlearrowleft$ and $\rightarrow$ are in parallel, as denoted by the $\wedge$ operator. Activity b is in loop, and can occur multiple times. The middle node is the re-do part of the loop and the right node of the loop is the exit node and it is executed exactly once. Hence, in the first child of the sequence operator $\rightarrow$, activity a can be repeated multiple times by performing activity f. After exiting the loop operator, activities e and c can occur in any order. Finally the process terminates after activity z is performed.

The usage of process trees in a genetic setting is motivated primarily by the fact that process trees are sound by definition. Hence, irrespective of the modifications performed, we would always end up with sound process models, which is desirable. That is, with process trees, we do not have to carry the unnecessary baggage of the incremental synthesis structure in a genetic setting, to guarantee soundness.

Of course, the use of process trees compared to the free-choice workflow nets has some limitations, as already discussed in Chapter 3. For example, process trees can only produce block-structured process models. Hence, discovering non-block-structured constructs, such as arbitrary loops, is difficult using process trees. However, we argue that the advantages of using process trees in a genetic setting outweigh the disadvantages. Moreover, by construction, any process tree can be easily translated

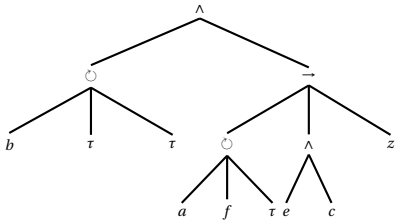
into a labeled free-choice workflow net [163]. However, it should be noted that the opposite could be cumbersome.

In the next section, we discuss the usage of genetic algorithm and the way in which constraints are verified in a process tree.

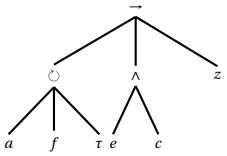
9.3.3 Genetic Algorithm

In order to repair the process tree to incorporate the user specified domain knowledge, we make use of a pre-existing genetic approach. We extend the genetic algorithm for process trees discussed in [25].

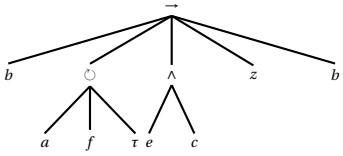
By default, the approach from [25] genetically produces process trees, which are then evaluated against the four standard quality dimensions of process mining, i.e., simplicity, generalization, fitness and precision. From Chapter 6, we know the intuition behind fitness and precision dimensions. The approach from [25] uses [5] for calculating these two dimensions. The simplicity dimension takes into account the complexity of the discovered process tree and the generalization dimension considers



(a) Example Process tree showing *sequence* (→), *and* (∧), and *Xor-loop* (↻) operators. This process tree corresponds to the labeled free-choice workflow net from Figure 9.2b (without ⊤ and ⊥ transitions).



(b) The process tree after removing the left-most ↻ node of the process tree from 9.6a. ∧ should have at least two child nodes, hence ∧ is also reduced away.



(c) The process tree after adding two instances of activity *b*.

Figure 9.6: Modifying process tree to obtain the process tree equivalent of the expected labeled free-choice workflow net from Figure 9.2a.

the amount of unseen possible future behavior that is allowed by a process tree.

We extend the approach from [25] to include an additional evaluator "number of constraints specified", along with the four standard quality dimensions. This evaluator considers the number of Declare constraints that are satisfied by a process tree.

We provide an intuition about how Declare constraints are verified using couple of examples. Consider two constraints, *chain-response(b,a)* and *chain-response(z,b)* corresponding to the process tree from Figure 9.6a. The constraint chain-response implies that the first activity is immediately followed by the second activity of the constraint. The common root operator between b and a is $\wedge$. This implies that b can happen both before a or after a. In other words, it cannot be guaranteed that a always happens immediately after b. A similar argument holds for the activities b and z and the constraint *chain-response(z,b)* in Figure 9.6a. Now, if we use the same constraints in the context of the process tree from Figure 9.6c, we see that the first b (left-most node) is guaranteed to happen immediately before the first a, as the common parent between a and b is sequence ($\rightarrow$), which is order preserving. Similarly, the second b is guaranteed to happen immediately after z. Hence the constraints *chain-response(b,a)* and *chain-response(z,b)* are satisfied in the process tree shown in Figure 9.6c.

A number of factors such as crossover and mutation probabilities, number of random trees in each generation etc. determine the variation of process trees in each generation. The tree modification process is guided to balance between the standard quality dimensions (determined by the event log) and the number of constraints satisfied. The genetic generation of process trees is stopped when the stopping criteria of the genetic algorithm are met. Stopping criteria could be the maximum number of generations, maximum time taken, minimum number of constraints satisfied etc.

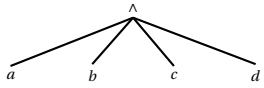
The end result is a Pareto front. The general idea of a Pareto front is that all models are mutually non-dominating: a model is dominating with respect to another model, if for all measurement dimensions it is at least equal or better and for one strictly better. Using the five dimensions, a Pareto front is presented to the user which contains the set of dominating process trees.

9.4 Implementation and Evaluation

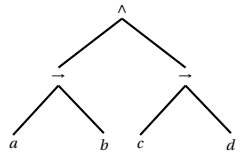
The first proposed technique (Section 9.2) for process model repair based on the engine from Chapter 5 is also available in the package *InteractiveProcessMiningLite 6.9.13* under the name "Reduce and synthesize Petri net with alignment and predictor view" supported by the nightly build version of of ProM [171]. This plug-in requires an event log in XES [3] format and a Petri net as its input. The second proposed approach (Section 9.3) is supported by the sand-box version of the package *PrabhakarDixit* under the plug-in "Genetic Modification with user constraints". This plug-in requires an event log in XES [3] format and a process tree as input.

We restrict the evaluation to the automated approach, as the interactive process model repair approach based on synthesis engine, is derived from, and hence is similar to, the interactive process discovery of Chapter 5.

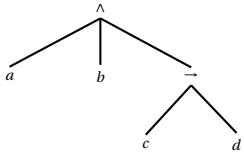
The automated process model repair approach is evaluated based on a synthetic



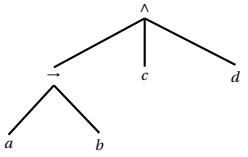
(a) Original process tree as discovered by Inductive Miner infrequent.



(b) One of the modified process trees in the Pareto front satisfying all the constraints .



(c) One of the modified process trees in the Pareto front satisfying only two constraints (*response*(c,d), *precedence*(c,d)).



(d) One of the modified process trees in the Pareto front satisfying only two constraints (*response*(a,b), *precedence*(a,b)).

Figure 9.7: Original and modified process trees for event log L.

event log and a real-life event log. Since we use the modified ETM algorithm as the genetic algorithm, some of the issues related to the ETM approach, such as long waiting times, also apply to our approach. We begin with the synthetic event log which essentially provides an intuition of things that may go wrong with automated process discovery techniques, and how these things can be corrected automatically by specifying domain knowledge using constraints.

9.4.1 Synthetic Event Log

We use a synthetic event log to demonstrate how our approach could improve an incorrect model discovered due to algorithm bias or noisy event log. For the event log $L = [\ \langle a,b,c,d\rangle^{90}, \langle a,c,b,d\rangle^{90}, \langle a,c,d,b\rangle^{90}, \langle c,a,d,b\rangle^{90}, \langle c,a,b,d\rangle^{90}, \langle c,d,a,b\rangle^{90}, \langle c,d,b,a\rangle^{6}, \langle c,b,a,d\rangle^{6}, \langle d,a,c,b\rangle^{6}]$, the Inductive Miner infrequent (IMi) [110] with default settings generates the process tree with all four activities in parallel as shown in Figure 9.7a.

From the highly frequent traces of the log we can deduce simple rules such as activity a is always eventually followed by activity b; and activity b is always preceded by activity a. Similar relationship holds for activities c and d.

We use this information to deduce the following four constraints: *response*(a,b), *precedence*(a,b), *response*(c,d) and *precedence*(c,d). We use these constraints, the process tree from Figure 9.7a discovered using IMi [110] and the synthetic log (L) as our input and perform the experiment with the genetic modification approach. We set the stopping condition of maximum generations to 500 generations. The population size in each generation is set to 100. Along with this, we add some additional conditions

restricting duplicate labels and guaranteeing a minimum number of nodes. All the other settings are kept as default as in the ETM miner of [25].

The modified process tree from Figure 9.7b satisfies all the four constraints. The tree from Figure 9.7b also has a higher precision value of 1, and a considerably high replay fitness score, compared to other process trees from the Pareto front. This process tree is highly precise, thereby explaining the high frequent traces of the event log much better and ignoring the infrequent noisy traces. Hence, if the user is interested the most in precision, then this is the process tree that would be chosen. However, if the user is flexible with respect to precision, then process trees from Figure 9.7c and Figure 9.7d might be interesting. These process trees have a very high precision score along with very high fitness scores, outperforming the originally discovered process tree. However, only half of the constraints specified are present in these process trees. Hence, if the user decides that some constraint(s) can be relaxed/ignored, then these process trees could be chosen by the user. Based on user's preferences, the Pareto front could be navigated to select the most interesting process models.

9.4.2 Real-life Event Log

During automated process discovery, *exceptional* cases may dominate the normal cases, thereby leading to a process model that is over-fitting the data or that is too general to be of any value. This process model could however be improved by incorporating domain knowledge. In order to evaluate such a scenario, we use the following steps on a real-life event log containing the road traffic fine management process with 11 activities and 150,370 cases described in and publicly available at [52]:

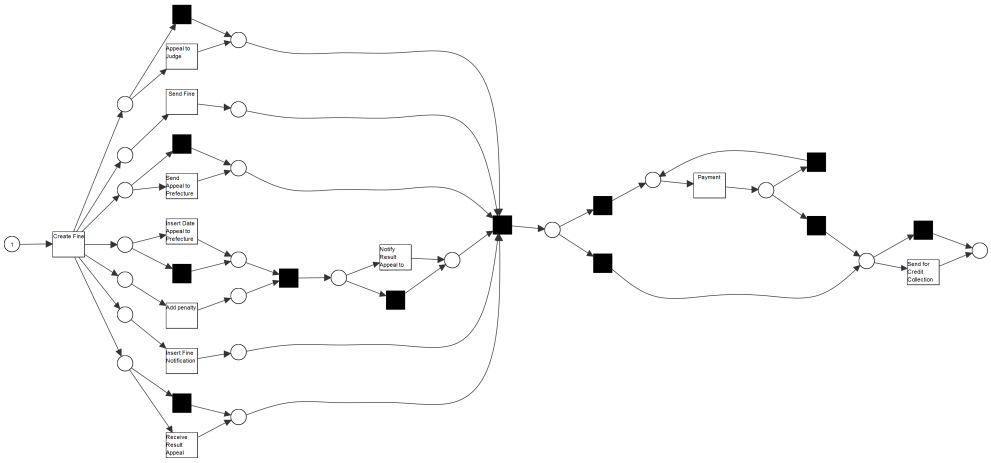


Figure 9.8: Workflow net mined using inductive miner infrequent [110] with filtered log containing infrequent traces only.

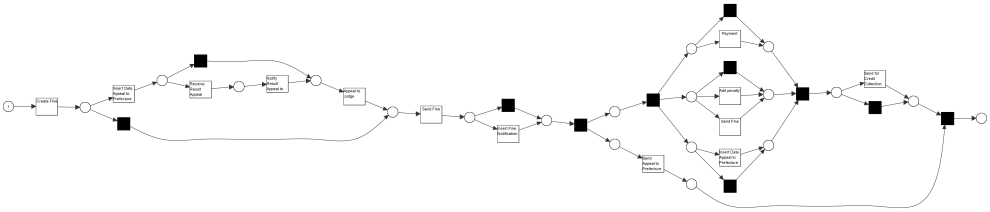


Figure 9.9: Labeled free-choice workflow net equivalent of one of the process trees repaired using the approach from Section 9.3.

1. Learn some *domain rules* that should hold in the process model, based on the insights from [52].
2. Filter the event log to select exceptionally deviating cases. That is, we filtered out the most common sequences of traces, and kept only those traces which are uncommon.
3. Create a process tree based on the filtered log using inductive miner algorithm [110]. For the ease of understanding, we show the labeled free-choice workflow net equivalent of this process tree in Figure 9.8.
4. Use the domain rules learned in step 1, the process tree from the filtered log of step 3, in combination with the *complete* event log as the input to the ETM genetic process discovery technique [24].

We deduced 1 precedence, 1 response, 1 responded-existence and 1 not-succession rule based on the insights from [52]. In the genetic modification approach, we set the stopping condition of maximum generations to 500 generations with the population size set to 100. All the other settings are kept as default from the ETM miner of [25]. Therefore, in total a maximum of 50000 trees are evaluated.

In order to evaluate the repair approach, we take one of the process trees discovered by the genetic approach with balanced quality dimensions satisfying all the constraints. The labeled free-choice workflow net equivalent of this process tree is shown in Figure 9.9. The chosen process model has a fitness score of 0.82 and a precision score of 0.74, when compared with the original (unfiltered) event log. However, the fitness and precision scores of the original process model, discovered using the filtered event log, was 0.72 and 0.54 respectively. Clearly, the repaired process model has a considerably higher quality score than the starting process model. Moreover, the repaired process model also satisfies all the constraints. It can be argued that similar results could have been achieved by tweaking some of the process discovery algorithms, such as the ILP algorithm [170], in order to include the domain knowledge. However, such techniques would only produce a single output process model. The genetic approach on the other hand, gives multiple such process models as the output. Therefore, the user can choose the most effective process model, in terms of the quality criteria and constraints satisfied, as well as in terms of comprehensibility.

9.5 Related Work

Conformance techniques in process mining such as [4, 5, 50] replay event logs on the process model to check compliance, detect deviations and bottlenecks in the model. These techniques focus on verifying the conformance of event logs with a process model, but do not provide any way of incorporating domain knowledge to interactively repair or improve the process model.

[136] provides a backward compliance checking technique using alignments wherein the user provides compliance rules as Petri nets. The focus is on using Petri net rules for diagnostic analysis on the event log, rather than discovering a new process model with compliance rules.

The conformance-based repair technique suggested by [78] takes a process model and an event log as input, and outputs a repaired process model based on the event log. This is similar to the automated approach proposed by us. However, our approach also allows the user to specify domain knowledge as a set of constraints, and this is included while repairing the process models. Moreover, the outcome of our approach could be more than one process model, where each process model would have different quality scores.

[8] presents an approach to interactively and incrementally repair a process model, based on the information from the event log. This is similar to the interactive approach proposed by us for repairing process models. However, our approach guarantees that the process models discovered are always sound, whereas [8] does not provide any such guarantees.

[84] and [144] propose approaches to verify the presence/absence of constraints in the process model. However, these approaches do not provide a way to modify the process models with respect to the constraints. In [116], the authors provide a way to mine declarative rules and models in terms of LTL formulas based on event logs. Similarly, [42] uses Declare taxonomy of constraints for defining artful processes expressed through regular expressions. [108] uses event logs to discover process models using Inductive Logic Programming represented in terms of a sub-set of SCIFF [6] language, used to classify a trace as compliant or non-compliant. In [40], the authors extend this work to express the discovered model in terms of Declare model. However, these approaches focus on discovering rules/constraints from event log without allowing the users to introduce domain knowledge during discovery/repair.

In [139], authors suggest an approach to discover a control flow model based on event logs and prior knowledge specified in terms of augmented Information Control Nets (ICN). Our approach mainly differs in the aspect of gathering domain knowledge. Although declarative templates can also be used to construct a network of related activities (similar to ICN), it can also be used to provide a set of independent pairwise constraints. [191] proposes a discovery algorithm using Integer Linear Programming based on the theory of regions, which can be extended with a limited set of user-specified constraints during process discovery. In [86], the authors introduce a process discovery technique presented as a multi-relational classification problem on event logs. Their approach is supplemented by Artificially Generated Negative Events (AGNEs), with a possibility to include prior knowledge (for example, causal depen-

dencies, parallelism) during discovery. The authors of [90] incorporate both positive and negative constraints during process discovery to discover C-net models. Compared to [86], [90], and [191], our approach differs mainly in the fact that we do not propose a new process discovery algorithm, but provide a generic approach to post process an already discovered process model, either interactively or automatically.

9.6 Conclusion and Future Work

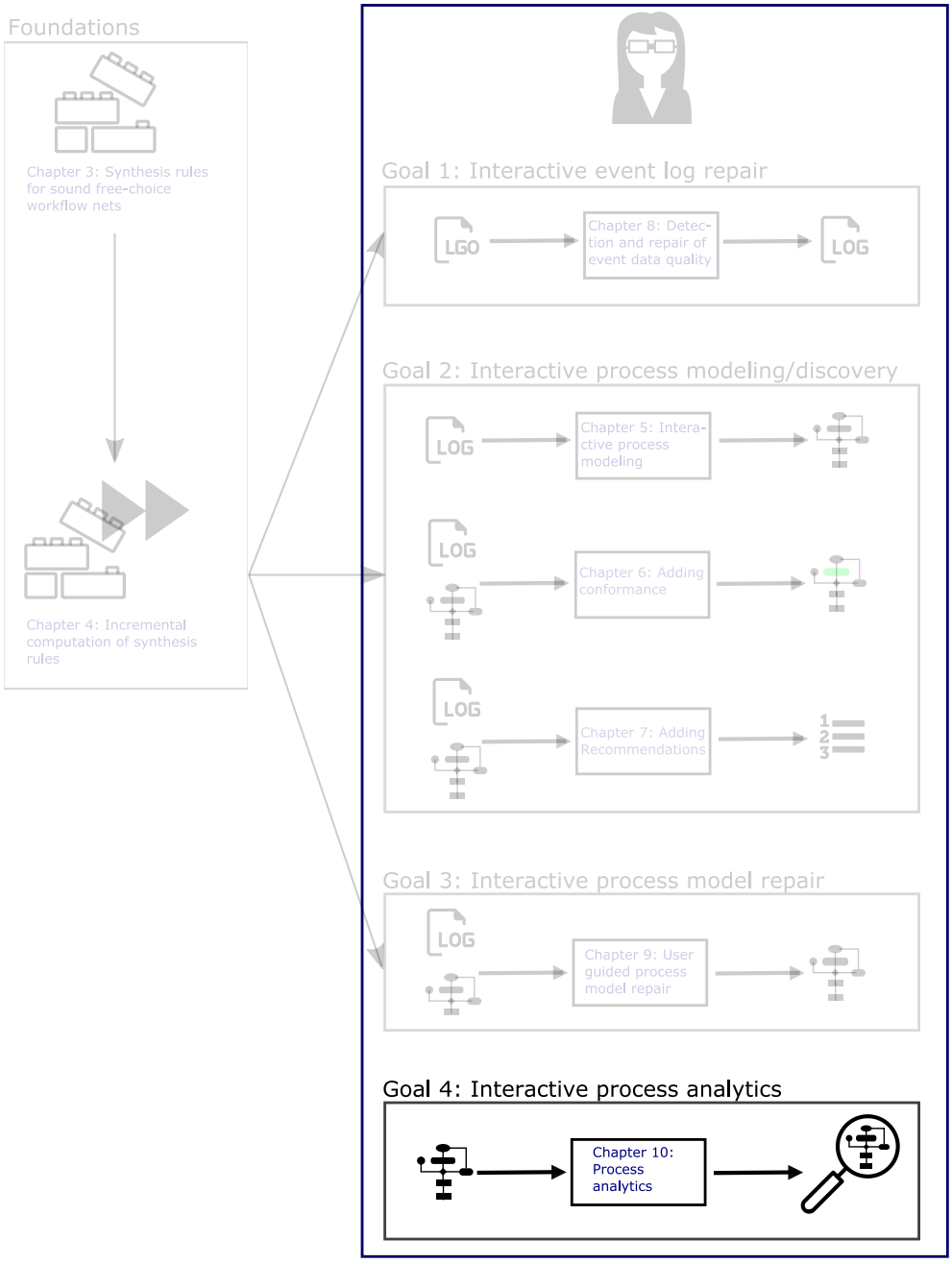
In this chapter we introduced a way to incorporate the domain knowledge of the expert to repair a process model. We primarily introduced two types of techniques: (i) the first technique allows interactive process model repair by intuitively considering domain knowledge; (ii) the second technique allows specification of domain knowledge with the help of constraints, which are used (along with event logs) to genetically discover multiple process models. The outcome of the second approach is a Pareto front of process models, balancing different quality dimensions of process mining along with the number of constraints satisfied.

For the first approach, it would be ideal to provide the user with smart recommendations in order to semi-automatically repair a process model. This relates to the approach discussed in Chapter 7. In Chapter 7, we provided recommendations which resulted in expansion of process models. However, in order to provide recommendations for interactive process model repair, we would also like to reduce a process model, and then subsequently expand it. Hence, in the future, it would be ideal to devise techniques that provide smart recommendations to assist the user in interactive process model repair.

For the second approach, we used process trees as the modeling notation to genetically modify and repair process models, owing to the structural properties of the process tree. In the future, it would be ideal to explore ways in which sound free-choice workflow nets could be used in a genetic setting, to overcome the limitations of process trees. Furthermore, the genetic approach could have very long waiting times, depending on the configuration of the system. Hence, it would also be viable to explore other heuristic-based approaches that could efficiently result in multiple process models as the output.

Chapter 10

User-driven Process Analytics



This chapter is based on the publication [64].

The execution histories of processes are recorded in information systems, and can be extracted using event logs. The extracted event data may contain some timestamp-oriented data quality issues, which can be overcome with the help of domain knowledge, using the interactive event log repair approach discussed in Chapter 8. Once an event log is deemed suitable for process mining analysis, one may interactively discover a process model based on the interactive process discovery techniques proposed in Chapter 3–Chapter 7. In cases when a process model already exists, it can be repaired to match with the reality as depicted by the event log using the interactive process model repair technique proposed in Chapter 9.

As discussed already, we use labeled free-choice workflow nets in combination with synthesis rules for representing process models, due to the advantages it offers, such as broad structural representations and guaranteed soundness. If the (possibly repaired) event logs extracted from the information systems are considered to be an accurate depiction of the reality, and if the (possibly discovered/repaired) labeled free-choice workflow net defines the ideal process flow, then the analysis performed by the conformance techniques can provide valuable insights about *what went right* and *what went wrong* in the process. Furthermore, the source of event logs, i.e., the information systems, may also record other *useful* information about a particular event and the corresponding case. That is, each event may contain additional event specific *data attributes*, for example, the *resource* which performed the particular activity represented by the event. Similarly, each case may have case-level data attributes. Table 10.1 shows an example of an event log with such attributes. Hence, the user can use the outcomes of Chapter 3–Chapter 9, to interactively explore the expected behavior of a process as depicted by the labeled free-choice workflow net with the reality as depicted by the event log, based on the conformance analysis.

In Chapter 6, we introduced a technique to speed-up conformance checking to support interactive process discovery. However, traditional conformance analysis techniques can be used to analyze the actual behavior of the process in reality, especially with an emphasis on local deviations and bottlenecks. Conformance analysis can be used to gain insights about the compliance aspect of the process, such as understanding where the deviations occur in a process [5, 129]. As there is a time notion associated with every step in a process, the conformance analysis could also be used in order to investigate the performance aspect of the process, for example, understanding where the bottlenecks occur in a process. In essence, the outcomes of conformance analysis could be combined with additional case and event-level data attributes, to interactively explore and analyze the *health* of a process in reality.

Commercial process mining tools, such as Fluxicon Disco [94] and Celonis Process Mining [1], allow interactive exploration of event logs and process models. Such commercial tools are especially useful for enhancing the event logs and process models with case and event-level attribute data. Furthermore, such tools allow easy filtering possibilities, and slider-based process model exploration, to easily discover different variants of process models. However, the semantics used for representing the process models are not always clear. Hence, it is not possible to perform conformance analysis on such process models, as the class of process models does not give any guarantees with regards to soundness.

Table 10.1: Example event log with attributes in a hospital setting.

CaseID	Activity	OrderDt	Module	Goals Met?	Age
182192	ZZOGLV1	2015-06-01	Module optimalisatie	Yes	70
182192	ZZOGLV1	2015-07-13	Module optimalisatie	Yes	70
182192	ZZOGLV2	2015-07-13	Module optimalisatie	Yes	70
182192	ZZOGLD2	2015-07-13	Module optimalisatie	Yes	70
182192	ZZOGLV2	2015-08-25	Module optimalisatie	Yes	70
182192	ZZOGLD2	2015-08-25	Module optimalisatie	Yes	70
182192	ZZOGLD4	2015-09-26	Module optimalisatie	Yes	70
182192	ZZOGLV4	2015-09-26	Module optimalisatie	Yes	70
182192	ZZOGLV5	2015-09-26	Module optimalisatie	Yes	70
182192	ZZOGLD5	2015-10-01	Module optimalisatie	Yes	70
182192	ZZOGLD4	2015-10-01	Module optimalisatie	Yes	70
198283	ZZOVA1	2015-01-22	Module overname	NA	40
198283	ZZOVV2	2015-01-22	Module overname	NA	40
198283	ZZOVD3	2015-02-11	Module overname	NA	40
198283	ZZOVA4	2015-03-12	Module overname	NA	40
198283	ZZOGLV1	2015-04-29	Module optimalisatie	NA	40
198283	ZZOGLD2	2015-04-29	Module optimalisatie	NA	40
198283	ZZOGLV2	2015-05-09	Module optimalisatie	NA	40
⋮	⋮	⋮	⋮	⋮	

On the other hand, in an academic setting, the applicability of visual analytics in process mining has been explored in [105, 118, 164]. Most of these visualizations display the conformance results in their entirety. From a user's perspective, analyzing the conformance results on a big and complicated process model with many nodes may turn out be too daunting. Moreover, most of the current techniques are standalone techniques and do not support effective combinations of inter-disciplinary analysis. Furthermore, investigation of the satisfaction of specific protocols and/or KPIs in the process is usually only possible through data-specific analysis [111]. That is, it is not possible to interactively perform various compliance and performance analysis directly on a process model.

In this chapter, we discuss ways to enable interactive process analytics, by using the outcomes of Chapter 3–Chapter 9. As opposed to traditional approaches, which usually provide metrics represented by some *numbers* which may be difficult for the user to comprehend, in this chapter we focus on the following types of enhancements [145]:

- **Helicopter views**: To enable users to gain a global perspective, the information from the event logs can be projected on the process model based on the outcome of alignment-based conformance technique, in order to provide insights such as frequencies of activities in the process etc.

- **Compliance analysis**: An aim of the analysis might be to identify if some of the protocols were followed in reality. Therefore, we provide a way for the user to interact with the process model to intuitively explore some of the possible compliance-oriented issues.
- **Root-cause analysis**: Case and event-level attributes could be used to possibly explain why certain deviations or bottlenecks occur in the process. That is, the user can visualize and quantify the information with the help of graphs, that can be used to answer some questions based on data attributes.
- **Concept drift analysis**: The user may be interested in finding out if there have been any changes in the process over time. In order to enable this, we provide a graph-based approach that can be used to explore (any possible) patterns over time.

In Section 10.1, we discuss the overview of the proposed system, and also show the application of helicopter views. In Section 10.1.2, we discuss the interactive aspect of the proposed system in detail, in order to enable compliance analysis, root-cause analysis and concept drift analysis. We discuss the conclusion and future research directions in Section 10.2.

10.1 Process Analytics

The proposed approach is supported by the nightly build version of ProM [171], in the *ProcessAnalytics* package, under the plug-in "Process Analytics". This plug-in requires an event log in XES [3] format and a labeled free-choice workflow net as its input. Furthermore, an alignment between the event log and the labeled free-choice workflow net is pre-computed based on an alignment-based conformance checking technique [5]. Figure 10.1 shows the snapshot of our tool, indicating the three main components.

Before discussing the overview of the system, we provide an overview of a running example used throughout this chapter. The event log used for analysis is a real-life dataset from a diabetes treatment care process, provided by a Dutch hospital. In order to better structure the diabetes treatment process [81], the hospital had proposed and rolled-out six specialized modules as a part of its diabetes treatment plan process. Each module can be viewed as a separate process, each consisting of a series of appointments, some of which may be skipped and some of which may occur in any order. Table 10.1 shows a snapshot of the event log. The event log contains data of over 2.5 years, and involves almost 300 cases (patients). Rather than splitting the event log into sublogs corresponding to each module type, we use the complete event log and used the interactive approach proposed in Chapter 7 to discover a process model, containing all the modules together.

We begin by discussing the process view, followed by the graph view. We discuss and show the usage of the configuration view in Section 10.1.2, in order to perform interactive compliance analysis, root-cause analysis and concept drift analysis.

10.1.1 Process View

In this section, we describe the first part of the interface, the so-called process view. As the name suggests, the process view is used to visualize the labeled free-choice workflow net. Furthermore, the labeled free-choice workflow net is projected with information from the event log, based on the outcome of the conformance analysis. In essence, the information projected could result in three types of helicopter views on the process. The user can switch between these three types using the radio-buttons, as shown in Figure 10.2. We now discuss each type of the helicopter view in detail.

Frequency View

Frequency view allows the user to get an overview of the actual occurrence frequencies of the overall process (for example, Figure 10.3). The frequencies of occurrence of an activity are obtained from the alignments projected on the process model. The frequency view encodes the frequency of occurrence of activities in the shades of blue, as projected on the synchronous moves in the alignments. Darker blue and lighter blue colors represent extremely frequent activities and infrequent activities respectively.

The user can easily identify the common and uncommon activities or fragments in the process, based on the background color of the activities in the process model. The following options are available to configure the view on frequencies, which can be chosen using the left-most drop-down menu shown in Figure 10.2.

- **Absolute View:** Calculates the frequencies, based on the absolute numbers.

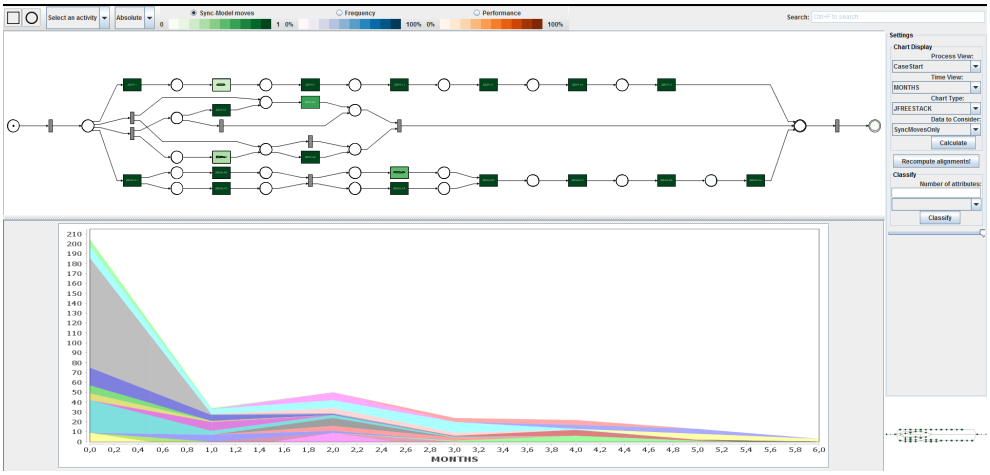


Figure 10.1: A snapshot of our interface. The top-most panel is dedicated to the process views. The user can directly interact with the process model. The visualizations (called graph view), in the bottom-most panel, are triggered by user interaction based on the configuration settings present in the panel on the right-hand side.

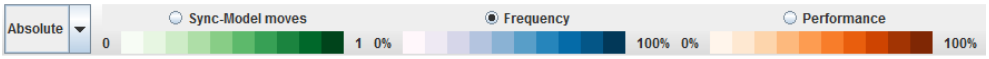


Figure 10.2: Radio-buttons for choosing helicopter views. The drop-down menu (left-side) becomes activated upon selecting frequency view.

For example, the color of an activity is determined by the absolute number of synchronous moves. The activity with most number of synchronous moves is colored the darkest. This view is similar to the default view of the inductive visual miner [111].

- **Average Occurrence per Case View:** Calculates the frequencies, based on the average occurrences per case. The color of the activity is determined by the number of times the activity has a synchronous move in a case, normalized over the event log. Hence, in case of a loop, if the activity has many synchronous moves within a trace, for many cases in the event log, then this activity would be colored the darkest. Similarly, if an activity has no synchronous moves (i.e., no corresponding event in the event log), then it would be colored the lightest.

- **Number of Cases:** Calculates the frequencies based on the number of cases for which the synchronous move occurred. The color of an activity is determined by the ratio of the number of cases for which synchronous moves occurred over the total number of cases in the event log.

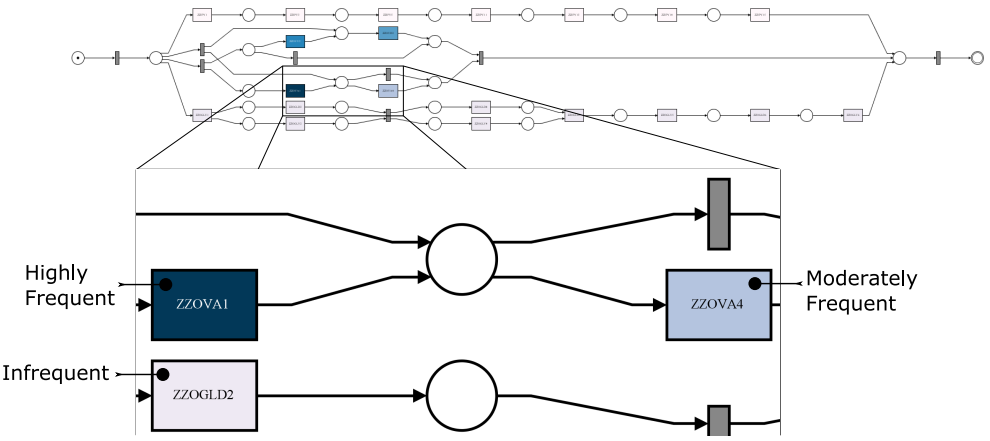


Figure 10.3: An example of frequency view of the process model. The more frequent activities could be easily visualized in the model.

Synchronous-Model-Log Moves View

The synchronous-model-log moves view is derived based on the synchronous and model moves, along with the log moves in the process model. This view represents the fitness of the event log and the process model. The activities are colored in the shades of green. The actual color of an activity is dependent on the ratio of:

$$\frac{\#Synchronous\ moves}{\#Model\ Moves + \#Synchronous\ moves + \#Log\ moves}$$

Hence, a darker shade of green for an activity in a model implies that the particular activity had more synchronous moves than model moves, i.e., it is described very well according to the model. The process view in Figure 10.1 shows an example of the synchronous-model-log moves view.

Performance View

The performance view allows the user to investigate the stages of the process where bottlenecks occur, or where there is room for improvement. In the performance view, the activities are colored in shades of red. The shade depends on the execution time between the immediate previous synchronous activity and the current synchronous activity for every case. A darker shade of red implies that the corresponding activity was executed after a long delay since the completion of the previous activity.

10.1.2 Graph View

In this section, we discuss the so-called *graph view*. This view is represented in the bottom-half of the snapshot of the tool shown in Figure 10.1. The contents of the graph view are based on the settings made, through the right-most panel of the tool as shown in Figure 10.1. The event data can be visualized in terms of stacked area charts and stacked bar charts, chosen via the *Chart Type* drop-down menu in the settings menu, as shown in Figure 10.4. We chose this representation because it makes comparisons more natural for the user and allows rapid discovery of outliers. We now discuss the contents of the settings panel, in order to obtain different graph views.

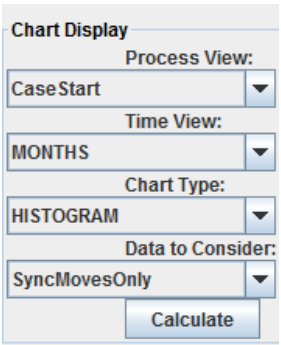


Figure 10.4: Settings for configuring the content of the graph view.

The X-axis (domain axis) of the graph view typically describes the time distribution and the Y-axis describes the frequency of occurrence. The X-axis can be sorted and configured using the *Process View* drop-down menu of the Chart Display settings panel as shown in Figure 10.4. Process View contains three choices, (i) Case Start, where the distribution of events is plotted in comparison to the start of a case, i.e., the first event of a case, (ii) Normal, where the distribution of events is plotted in comparison

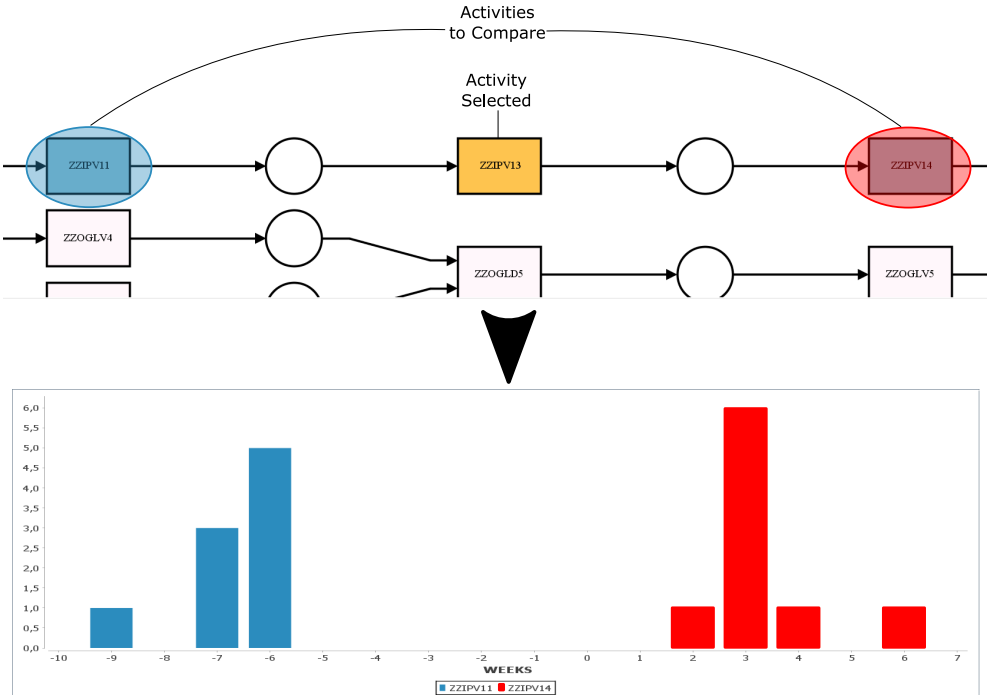


Figure 10.5: Graph view to compare the distribution of activities with respect to a selected activity. A single activity from the process view is chosen by the user as the base activity, followed by a number of activities which should be compared with the base activity. The histogram shows the time distribution and the number of times the selected activities to be compared occur, *before* or *after* the base activity. The color of each bar in the histogram corresponds to each selected activity from the process, except the base activity. It should be noted that the base activity always occurs at time 0 and is not shown in the histogram view.

to the first event in the event log, (iii) Activity, where the distribution of events is plotted in comparison to the occurrence of an activity selected in the process model by the user.

Furthermore, it is possible to abstract the absolute timescale into categories such as: the day of the week when the event occurred, or the month of the year when the event occurred etc. using the *Time View* drop-down menu from the Chart Display panel as shown in Figure 10.4.

The X-axis of the graph view represents the configurable absolute or relative timescales for the user. The Y-axis, i.e., the frequency axis is also configurable. The user can choose from plotting only synchronous moves, only log moves, or both the synchronous and log moves from the alignment. Viewing both synchronous and log moves essentially shows the information from the complete event log. This choice is configurable using the *Data to Consider* drop-down menu from the Chart Display

panel as shown in Figure 10.4. In the next sections, we show how these settings could be configured in order to interactively address the enhancement tasks of performing compliance analysis, root-cause analysis and concept drift analysis.

10.1.3 Compliance Analysis

In this section, we discuss how we can perform generic compliance analysis using the interactive capabilities of the system to detect possible compliance issues that might exist in the process. For more sophisticated compliance analysis, we refer to specific techniques by [104, 137]. The performance view allows the user to easily investigate the overall time between activities and/or where the bottlenecks may be in the process. However, there could be some time-critical KPI analysis that the user might be interested in. For example, a certain activity 'B' should be performed within *x-hours* after executing an activity 'A'. The user can select the activity 'A', and activity 'B' (among others if needed), to visualize the time span of occurrence distributions of 'B' with respect to 'A'.

Figure 10.5 shows an example of performing such type of conformance analysis. The chosen activity (i.e., the appointment code ZZIPV13), should ideally be completed within 9-10 weeks after the appointment ZZIPV11. Similarly, the appointment ZZIPV14 should also be completed within 9-10 weeks after completion of the appointment ZZIPV13. Clearly, both these protocols are followed in reality according to the event log, as shown in Figure 10.5.

10.1.4 Root-cause analysis

The *Synchronous-model-log moves view* provides a good overview of how well the data and model fit, and where the possible deviations are located in the process. The next step would be to investigate what causes deviations in the process. For example, suppose that the conformance analysis indicates that a particular task is sometimes

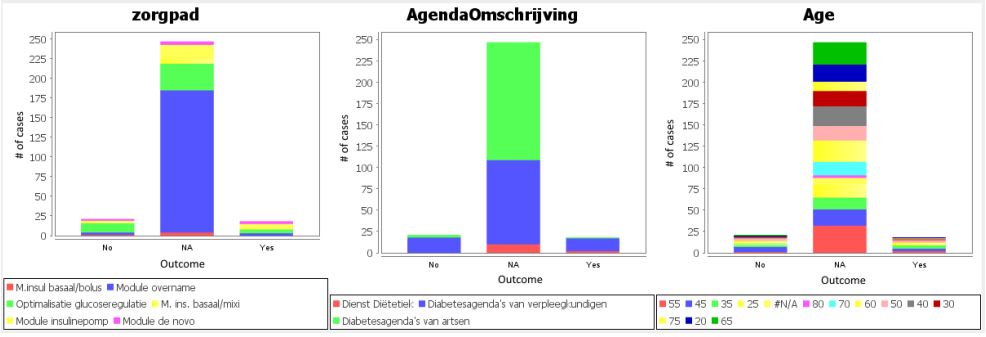


Figure 10.6: The top-three-ranked attribute classifiers for the classification analysis based on the answers to the question *Were the personal goals met?* The top-ranked attribute classifier based on the ranking algorithm was zorgpad (i.e., *module type*).

skipped (move on model) in the process. This could be explained by the attributes associated with the case, for example, the resources performing the task.

In order to investigate what or who causes deviations in the process, classification analysis is used. Traditionally, classification techniques in process mining context are used to perform tasks such as abstracting event logs, identifying bottlenecks, understanding and modeling guards in data-aware process models, performing correlation analysis, annotating and clustering cohorts by splitting event logs into sublogs etc. [23, 51, 85, 135]. In our approach, we use classification techniques from [97] in order to perform root-cause analysis. In particular, we support the use of pre-existing feature selection techniques, which use data attribute values (either case-level attributes or event-level attributes) in order to classify based on the desired output. Furthermore, well-known ranking techniques are used to determine the rank of each attribute based on the ability of an attribute to classify based on the desired output. The classification settings are also present in the Settings panel. The user can select the number of attributes *n* to be considered, and the type of classifier to be used in the Classify panel, as shown in Figure 10.7. Based on this, the *top-n* ranked attributes are shown to the user which best classify the desired output. The desired output could be an activity from the process model, or a case-level attribute.

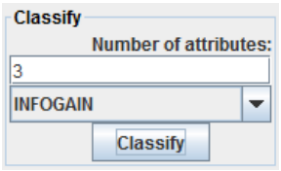


Figure 10.7: Classify panel to configure the classification options.

For our running example, at the end of each module, the patients are asked to fill in a questionnaire, evaluating the patient-treatment plan, understanding and satisfaction of the module. We consider one such question: *Were the personal goals met?*.

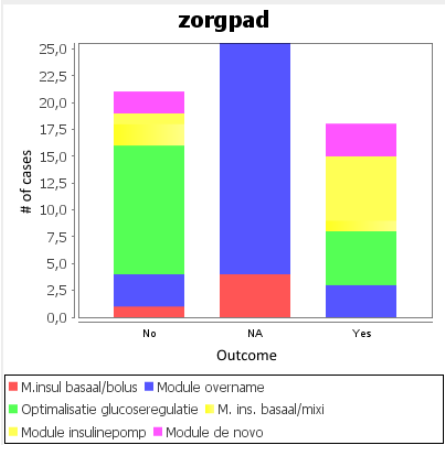


Figure 10.8: Zoomed in version showing responses for Yes and No, for the top ranked classifier zorgpad (*module type*) based on the answer to the question *Were the personal goals met?*.

The possible outcomes of this question (*Yes, No or NA*) are modeled as the optional ending activities the process model. Note that all the patients have at least one of these values for this questions, and therefore at least one synchronous move for the three optionally modeled activities. Next, we performed feature selection analysis with the Information gain classifier, and synchronous moves to the answers to question (*Yes, No or NA*) as the output, as shown in Table 10.1. The top-three ranked attributes which best classify the output are shown in Figure 10.6. It is however important to note that *module overname* is a basic module (to meet diabetic team members) and hence the majority of the patients from such modules have not reached their goals yet, thereby having the corresponding value of NA, as shown in Figure 10.6.

We selected the top-ranked attribute classifier, found to be the *module type*. For the majority of patients, no value was recorded for this question (value of NA). However, for the patients who did fill in the survey, one prominent outcome, as evident from Figure 10.8 was that for the module *Optimalisatie glucoseregulatie*, almost twice the amount of patients did not meet their goals fully. This suggests that another module might have been more suitable or calls for improvement in the module to better manage the patient expectation.

10.1.5 Concept-drift Analysis

Our tool also supports ways of exploring possible concept drift issues. For our running example, by plotting the data for all the events, of all the modules, its easy to visualize the patterns of changes in the types of module over time. The domain axis was configured to indicate the *Normal* timescale (i.e., the absolute timescale starting from the first event to the last event in the event log), represented in terms of months.

From Figure 10.9, it can be easily concluded that for the first few months (approximately until 7 months), only one module existed, as all the activities within this time span belong to the same module type. From Figure 10.9 it can also be seen that this module still persisted over time, but was gradually replaced by the other modules.

One more interesting pattern that could be observed from Figure 10.9, is the decrease in the number of events towards the end of the timeline. A logical explanation for this pattern could be that since the appointments (events) for modules are usually scheduled in the future, a more distant future has fewer number of appointments.

10.2 Conclusion

In this chapter, we introduced techniques for enabling interactive process-oriented data analysis. The proposed techniques build upon and bring together existing techniques from process mining, data mining and visual analytics field, to enable interactive process analysis. It supports exploratory analysis through different *helicopter* views on the process. In contrast to existing approaches, it is highly interactive. Furthermore, the tool supports possibilities to intuitively perform compliance analysis, root-cause analysis and concept drift analysis.

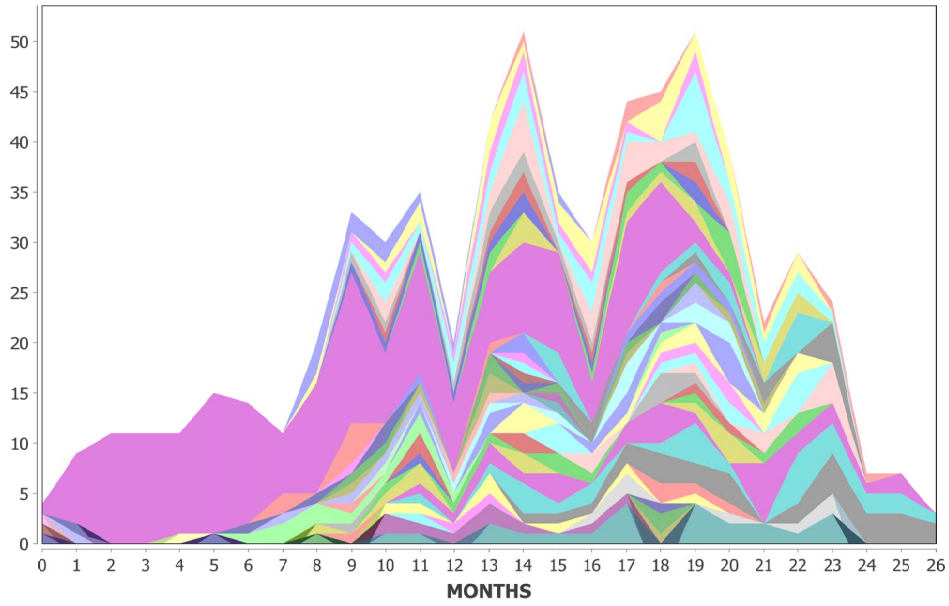


Figure 10.9: Graph showing the concept drift and changes among different modules in the LUMC dataset. The colors in the graph correspond to an activity belonging to a particular module. For the first 7 months, all the activities belonged to one module type. There is a steep fall in the module usage towards year end and a peak in the module usage in year 2.

As a part of future work, drill-down and roll-up analysis could be introduced, to support more data-oriented tasks, by keeping the labeled free-choice workflow nets at the core. The application of root-cause analysis could be further improved by considering the combined impact of multiple attributes on the output variable for classification, as currently we only considered one attribute at a time. Here, statistical analysis would also be a beneficial addition.

Chapter 11

Conclusion

This chapter concludes the thesis. We begin by summarizing the main contributions of this thesis (Section 11.1), followed by a reflection on the limitations (Section 11.2) and a discussion on future research directions (Section 11.3).

11.1 Contributions

In this thesis, the focus has been on the intersection of the traditional human-driven process modeling setting and data-driven process mining setting. With this as the aim, we introduced four research goals, which provide techniques that enable human-in-the-loop process mining, as shown in the overview in Figure 11.1.

11.1.1 Foundations

Before addressing the four goals of interactive process mining, we first discussed the foundations that were required to enable these goals. In particular, we looked at Goal 2, i.e., interactive process discovery, and the subgoals as discussed in Section 1.4.2 of Chapter 1. The first subgoal of Goal 2 required that the modeling notation used to represent the process models should represent both the simple and complex structural constructs, such as sequences, choices, loops, concurrency, arbitrary choices and non-block-structuredness. In order to address this subgoal, we chose the class of free-choice workflow nets in our approach, as it could be used to represent broad structural constructs. The second subgoal required a guarantee that any process model discovered by the user should be a sound process model. In order to address this, we used the synthesis rules proposed in [58], in the context of free-choice workflow nets.

In Chapter 3, we built upon the synthesis rules for free-choice nets [58] in order to deduce synthesis rules for free-choice workflow nets. Furthermore, we showed that the derived rules for free-choice workflow nets preserve soundness [168] property of workflow nets, if we start with a sound atomic free-choice workflow net. We ended

Chapter 3 by showing that even though the synthesis rules for free-choice workflow nets could serve as the basis for a synthesis engine, it is inefficient to compute all the applications synthesis rules for a free-choice workflow net in a brute-force way.

The brute-force technique proposed in Chapter 3 was inefficient and hence im-

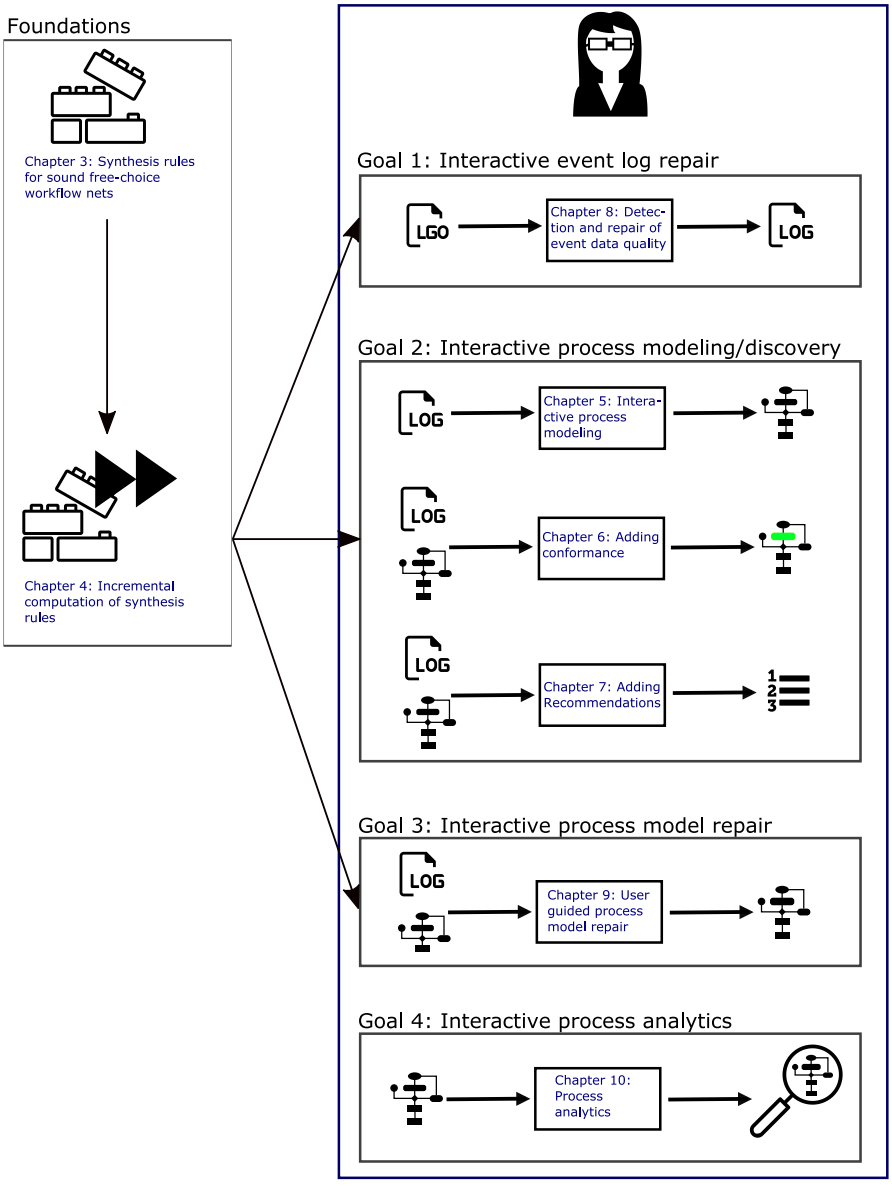


Figure 11.1: Overview of the thesis.

practical in an interactive setting [127]. In Chapter 4, we addressed the challenge posed in Chapter 3 for computing all the possible applications of synthesis rules for a free-choice workflow net in an efficient way. In particular, we proposed an incremental way of computing synthesis rules. Starting with an atomic net, and an intermediary structure, we proposed an approach to incrementally compute all the possible applications of synthesis rules using the intermediary structure. The intermediary structure could then be updated based on the synthesis rule used, in order to obtain the information which could be used for computing all the applicable synthesis rules corresponding to the newly synthesized net. Furthermore, we showed that the approach for calculating the synthesis rules from the intermediary structure is correct, i.e., it is possible to correctly extract only possible applications of synthesis rules. Moreover, we also showed that the approach is complete, i.e., by using the approach it is possible to deduce any possible sound free-choice workflow net starting with an atomic net by using the synthesis rules derived from the intermediary structures.

11.1.2 Interactive Process Modeling

The foundations discussed in Chapter 3 and Chapter 4 resulted in a synthesis engine, which could be used to synthesize any sound free-choice workflow net. The next subgoal of interactive process modeling/discovery, dealt with building an interactive editing engine which allowed editing of process models in an incremental and structured manner [43]. In Chapter 5, we first introduced the concept of labeled free-choice workflow nets in order to label the transitions from the free-choice workflow nets with activity names. Next, we proposed a way to use the synthesis rules (from the synthesis engine), in order to interactively construct a labeled free-choice workflow net, starting from the initial atomic net, thereby addressing the subgoal of incremental and structured editing of process models. The next subgoal required a way to extract and present the information from the event logs to the user for better decision-making. We used an approach similar to the one used by some of the automated process discovery techniques by abstracting the event log information, and presenting it to the user in raw format, as well as by projecting it on the labeled free-choice workflow net based on the activity selected. The proposed approach was evaluated using a case study and an objective evaluation based on the process discovery contest.

In Chapter 6, we discussed strategies to support the subgoal of conformance analysis. The idea here was to improve the data support for providing the users with an intuition about the impact of a change made to a process model, by providing active conformance support. As the traditional conformance approaches were deemed *too slow* in certain scenarios, we came up with a technique to compute approximated conformance scores in a fast-enough way, that is suitable in an interactive setting. Thus, the user is provided with an indication about an impact of a change made to a process model. The proposed conformance technique was evaluated against multiple real-life event logs to show the effectiveness, both in terms of speed and accuracy, compared to the state-of-the-art conformance analysis techniques.

In Chapter 7, we moved towards automating process modeling tasks on behalf of the user, i.e., semi-automated process discovery. This chapter mainly addressed the

subgoal of providing recommendations to the user. The main challenge here was to provide meaningful recommendations but in fast-enough way, acceptable in an interactive setting [127]. The proposed technique uses some pre-existing patterns to automatically provide recommendations for positioning of activities within a process model. The recommendations are ranked based on the conformance results introduced in Chapter 6. Moreover, the process model is changed minimally in order to display the positioning of the newly added activity. The user could thus decide on the desired level of complexity, and could incrementally and interactively discover process models. The proposed technique was evaluated via a user study.

11.1.3 Interactive Event Log Repair

In Chapter 8, we provided a one-stop solution to infuse user knowledge to correct data quality issues in the event log. Since the timestamp is central to all process mining analysis, we focused only on the ordering-related issues in the event log. We first identified three subgoals vital for enabling interactive event log repair, in a process mining setting, discussed in Section 1.4.1 of Chapter 1. The first subgoal dealt with identifying the possible issues in the event log. As the main aim is to repair the event log, we provided some automated techniques to detect possible ordering-related issues in the event log. The second subgoal was to interactively repair the event log. Since it is ideal to repair many instances of incorrect events at once, rather than on a case-by-base basis, we proposed a process fragments based approach for performing the event log repair. The process fragments were constructed based on the editor from Chapter 5. The third subgoal was concerned with providing an impact of the changes in the original (before repair) and the repaired event log. In order to address this, we calculated and presented the user with several statistics. The proposed techniques were evaluated using two real-life event logs.

11.1.4 Interactive Process Model Repair

In Chapter 9, we provided two approaches to interactively repair a pre-existing process model using the inputs from an event log, guided by the user. This relates to Goal 3 (Section 1.4.3) from Chapter 1. We first identified two ways in which user knowledge could be included in the process model, i.e., implicit or explicit. Hence, we proposed two approaches, to support both the ways sufficiently. The first approach used Chapter 5 and Chapter 6 as the basis in order to enable the user to interactively edit/repair a process model. In such cases, the domain knowledge would be implicitly considered by the user. The second approach allowed the user to specify domain knowledge in terms of certain constraints. The user-specified constraints were then combined with the information from the event log in order to genetically generate multiple process models for the user to choose from.

11.1.5 Interactive Process Analytics

Chapter 10 is essentially a tool paper, which combines the works of Chapter 3–Chapter 9, to interactively explore the execution of process in reality, by combining visual analytics with process mining. We proposed a system to interactively explore any conformance and/or performance issues in the process, in order to analyze the health of the process.

11.2 Limitations and Open Issues

We have provided techniques to enable interactive process mining at various levels. In this section, we discuss the main limitations of the proposed techniques and some of the open issues.

11.2.1 Representational Bias

One of the primary contributions of this thesis is enabling interactive process modeling/discovery. The synthesis rules used in the background guarantee that the user stays inside the realm of sound labeled free-choice workflow nets, and it is possible to discover any sound labeled free-choice workflow net starting with an atomic labeled free-choice workflow net. However, the sequence in which activities are added to the process model is extremely important. As the user is allowed to edit a process model in a certain way (guided by the synthesis rules), prior decisions made during process modeling could limit the construction of desired process models. That is, the decisions made in the past could severely restrict the possible process models discoverable in the future using the synthesis rules. A workaround this could be to go back and then forward again, i.e., use a combination of the reduction rules and the synthesis rules.

Next, only one activity can be added at a time to a process model. Moreover, complex constructs such as hierarchical process models are not supported by the tool. Another point of concern is that the synthesis rules used can only deduce free-choice constructs. Thereby, long-term dependencies that can be modeled easily using non-free-choice constructs can only be modeled by using duplicate activities in the process model. Having multiple occurrences of the same activity in the process model would increase the number of nodes in the graph, and hence eventually make the process model more complicated. A way to deal with this could be to post-process a free-choice net, to model the long-term dependencies as soft constrains.

11.2.2 Limited Data Usage

For the interactive process modeling/discovery, the user is supported with data from the event logs concerning the control-flow aspect only. However, the user may also be interested in using other data attributes while making decisions during the phase

of process modeling/discovery. Other data attributes are currently not supported in data projection and conformance analysis proposed in Chapter 5 and Chapter 6.

Similarly, for the goal of interactive event log repair, only part of the data is considered. The proposed approach focuses only on a single aspect, i.e., correcting the ordering of events in the event log. This mainly deals with the timestamps of events in the event log. However, as has been noted in the literature [151, 163], there are many more data quality issues that may exist in the event log, which are not necessarily ordering-related.

11.2.3 Lack of Guarantees

The recommendations proposed for semi-automated process discovery in Chapter 7 allow the users to switch between automated and manual process discovery. However, the automated component is constrained by some predefined patterns that are used to construct process models. These patterns are not complete, and do not give any guarantees with regards to the search space explored. Moreover, when multiple process models have similar conformance scores, it could be difficult to analyze the differences in the process models by merely looking at the conformance scores.

The proposed conformance framework is clearly beneficial in an interactive setting over traditional conformance analysis, especially in terms of computation times. However, as already indicated in Chapter 6, one of the primary limitations of the proposed approach is that it cannot distinguish the duplicate occurrences of activities in the process model. Thereby, when an activity is repeated in a process model, it is impossible to detect if a certain instance of activity in the process model is more conforming than the other. Clearly, such insights would be beneficial in providing local diagnostics. Moreover, duplicate activities can also lead to computation of incorrect fitness and precision scores, and hence give incorrect diagnostics.

11.2.4 Lack of User Evaluation

One of the more generic, yet prominent limitations, throughout this thesis is the lack of extensive user evaluations. As the thesis discusses techniques for adding interactive capabilities in process mining scenarios, it seems natural to perform thorough user evaluations. To this end, we have only performed a user study in Chapter 7 which uses an informal interview based approach at analyzing and detailing user behavior. However, since the number of users was limited to three users, we did not perform any statistical analysis.

11.3 Future Directions

In this section, we discuss some of the most promising future research ideas, based on the works presented in this thesis.

11.3.1 Improve Process Editing Engine

Since the approach proposed for interactive process modeling/discovery uses synthesis rules for free-choice nets as the basis, it has a limited expressiveness. Moreover, the usage of synthesis rules as a process modeling engine may result in scenarios where the user can no longer discover certain structures, due to some of the prior decisions made during process modeling. The workaround of going back and coming forward using a combination of reduction rules and synthesis rules could do the trick. However, this may be inconvenient, especially when the process model is large, and it is needed to navigate back to the initial few synthesis steps. Therefore, it would be ideal to build an engine that allows the construction of a broader class of process models, i.e., not restricted to free-choice constructs, while guaranteeing properties such as soundness. It is also imperative that the process editing engine also allows flexibility for the user to deduce any structure of the process model, from any given process model. Moreover, the process engine should also enable adding multiple activities at once and/or constructing hierarchical process models.

11.3.2 Enhance Data Usage

An event log contains a lot of information, in addition to the control-flow perspective. Many times, the additional information may be useful to the user while modeling process models. Hence, one future direction could be to combine and present additional information about event-level and case-level attributes, during the process of interactive process discovery/repair. Another future direction could be to repair the non-ordering related data quality issues in the event log. Therefore, it would be desirable to come up with techniques that enable interactive repair of event logs beyond timestamp ordering of events, for example, re-naming activities, abstracting activities and so on.

11.3.3 Improve Support

As noted in Chapter 6, using the current conformance analysis, it is not possible to distinguish between duplicate occurrences of activities in the process model. Hence, one future direction could be to address this in order to provide a more accurate localized indication when duplicates are present in the process model. Another future direction with regards to conformance could be to allow the users to specify certain constraints/rules that should hold in the process model. Thereby, active conformance analysis could be performed based on the pre-specified rules as well as the event log corresponding to the process that is being modeled interactively.

Another future direction could be to improve the recommendation support for assisting in semi-automated process discovery. Currently, the patterns used for deducing process models are not complete. Hence in the future, it would be ideal to provide a robust recommendation engine, that can provide guarantees regarding the class of discoverable models. Moreover, it would also be ideal to come up with additional metrics, other than fitness and precision values, that provide better intuition about

different recommendations in order to differentiate similarly ranked recommendations.

11.3.4 Getting User Feedback

The goal of the proposed interactive process mining techniques is to gain insights, for example, by discovering a graphical process model from an event log. However, defining and measuring insights is a challenging task which requires consideration of various parameters [132]. For example, in the BPM discipline there are multiple notations possible for representing process models. Hence users preferences for a particular notation can easily interfere while evaluating the interactivity in interactive process mining. In general, in this thesis, we primarily evaluated the advantages that interactivity brings when we have an expert user in the loop, compared to automated process mining techniques, in an objective way. Therefore, in the future, it would be interesting to explore different subjective parameters that affect interactivity, from different perspectives. The feedback could then be used to evolve and improve the proposed techniques.

11.3.5 Use Different Modeling Notations

In this thesis, we used free-choice workflow nets in combination with synthesis rules, in order to represent the process models. The motivation for choosing free-choice workflow nets and its advantages has already been discussed. However, in the future, it could also be ideal to use some other process modeling notations. For example, process trees are sound by construction, and can also discover many structural representations that a free-choice workflow net can. Clearly, there are notable differences between using process trees as the modeling notation and free-choice workflow nets with synthesis rules as the modeling notation in an interactive setting. For example, it would be possible to remove multiple nodes at once, by removing the root node from the process tree. However, in a free-choice workflow net, only a single place and/or a single transition could be removed (using the reduction rules). Hence, it could be ideal to use other modeling notations, such as process trees, in an interactive editing system, and compare the approaches. However, it should be noted that, some of the limitations of free-choice workflow nets, such as difficulty in handling long-term dependencies, also applies to process trees.

11.3.6 Online Setting

The techniques proposed in this thesis focused on enabling interactive process mining. However, the proposed techniques could easily be extended to other areas, such as online process mining. In such settings, an event log is not available beforehand. Instead, streams of events are made available, as they happen in reality. The amount of data generated could be too high to be stored locally. We could adapt some of the techniques proposed in this thesis to be applicable in an online setting. For example, consider the approach proposed for computing conformance in Chapter 6. If we have

a preexisting process model in an online setting, then the roles would be reversed. That is, in Chapter 6, we assumed that an event log remains the same, and the process model is changing. However, in an online setting, the process model would remain the same, and the event log would keep changing. Hence, we could compute new footprint patterns based on the incoming events, and compute the conformance on-the-go.

11.3.7 Robotic Process Automation

Another possible future direction could be to use interactive process mining approaches proposed in this thesis, in order to assist the deployment of Robotic Process Automation (RPA). In a nutshell, RPA allows automated execution of process oriented tasks, based on a pre-defined configuration by the user. In order to deploy an RPA solution, to automate a process, it is first necessary to know what goes on in the process. The interactive process discovery approach, presented in Chapter 5–Chapter 7, could thus be used in order to discover the process that needs to be automated. Furthermore, the process analytics system, proposed in Chapter 10, could be used to understand and improve the possible limitations of the process.

11.3.8 Assist Automated Process Discovery

Another generic future direction could be, to combine the approaches proposed in this thesis, with some preexisting automated process discovery techniques. For example, the conformance checking algorithm proposed in Chapter 6, is built around the assumption that the event log remains the same, while the process model changes. Genetic process discovery approaches have a similar setting. These techniques create hundreds of process models in each generation. Each of these process models have to be evaluated against the event log in order to get the conformance score of the process model. Since the event log is unchanged, we could also use the approach proposed in Chapter 6 to compute conformance scores in an efficient way in a genetic process discovery setting. Similarly, the constraints-based repair technique proposed in Chapter 9, could also be extended to other automated process discovery settings, for example the ILP miner. Pre-specifying the domain knowledge as constraints could assist the automated process discovery techniques in discovering better process models.

Bibliography

[1] Celonis. `http://www.celonis.com`. Accessed: 2019-01-20. (Cited on page 219.)

[2] Process discovery contest 2017. `https://www.win.tue.nl/ieeetfpm/doku.php?id=shared:process_discovery_contest`. Accessed: 2019-01-20. (Cited on page 101.)

[3] IEEE standard for eXtensible Event Stream (XES) for achieving interoperability in event logs and event streams. *IEEE Std 1849-2016*, pages 1–50, Nov 2016. (Cited on pages 101, 136, 161, 188, 211, and 221.)

[4] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst. Conformance checking using cost-based fitness analysis. In *Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International*, pages 55–64. IEEE, 2011. (Cited on pages 142 and 215.)

[5] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst. Towards Robust Conformance Checking. In *Business Process Management Workshops*, volume 66 of *Lecture Notes in Business Information Processing*, pages 122–133. Springer Berlin Heidelberg, 2011. (Cited on pages xviii, 17, 106, 109, 110, 136, 142, 184, 188, 194, 196, 210, 215, 219, and 221.)

[6] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Verifiable Agent Interaction in Abductive Logic Programming: The SCIFF Framework. *ACM Trans. Comput. Logic*, 9(4):29:1—-29:43, aug 2008. (Cited on pages 144 and 215.)

[7] M. Alizadeh, M. D. Leoni, and N. Zannone. Constructing probable explanations of nonconformity: A data-aware and history-based approach. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1358–1365, Dec 2015. (Cited on page 142.)

[8] A. Armas Cervantes, N. R. T. P. van Beest, M. La Rosa, M. Dumas, and L. García-Bañuelos. Interactive and Incremental Business Process Model Repair. In Hervé Panetto, C. D., W. Gaaloul, M. Papazoglou, A. Paschke, C. A. Ardagna, and R. Meersman, editors, *On the Move to Meaningful Internet Systems. OTM 2017*

Conferences, pages 53–74, Cham, 2017. Springer International Publishing. (Cited on pages 113 and 215.)

[9] A. Augusto, R. Conforti, M. Dumas, and M. La Rosa. Split Miner: Discovering Accurate and Simple Business Process Models from Event Logs. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1–10, nov 2017. (Cited on pages 89, 111, and 112.)

[10] Ahmed Awad, Rajeev Goré, Zhe Hou, James Thomson, and Matthias Weidlich. An iterative approach to synthesize business process templates from compliance rules. *Information Systems*, 37(8):714 – 736, 2012. Special Issue: Advanced Information Systems Engineering (CAiSE'11). (Cited on page 112.)

[11] E. Badouel, L. Bernardinello, and P. Darondeau. *Process Discovery*, pages 283–300. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. (Cited on page 47.)

[12] F. Belardinelli, A. Lomuscio, and F. Patrizi. Verification of gsm-based artifact-centric systems through finite abstraction. In C. Liu, H. Ludwig, F. Toumani, and Q. Yu, editors, *Service-Oriented Computing*, pages 17–31, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. (Cited on page 144.)

[13] E. Bellodi, F. Riguzzi, and E. Lamma. Probabilistic Declarative Process Mining. In Y. Bi and M. Williams, editors, *Knowledge Science, Engineering and Management*, pages 292–303, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. (Cited on page 111.)

[14] R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Process mining based on regions of languages. In *International Conference on Business Process Management*, pages 375–383. Springer, 2007. (Cited on pages 111 and 112.)

[15] R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Synthesis of Petri nets from finite partial languages. *Fundamenta Informaticae*, 88(4):437–468, 2008. (Cited on page 111.)

[16] R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Synthesis of Petri nets from scenarios with VipTool. In *International Conference on Applications and Theory of Petri Nets*, pages 388–398. Springer, 2008. (Cited on page 48.)

[17] J. Bernard, D. Sessler, T. May, T. Schlomm, D. Pehrke, and J. Kohlhammer. A visual-interactive system for prostate cancer cohort analysis. *IEEE Computer Graphics and Applications*, 35(3):44–55, May 2015. (Cited on page 196.)

[18] G. Berthelot. Transformations and decompositions of nets. *Petri Nets: Central Models and Their Properties*, pages 359–376, 1986. (Cited on page 46.)

[19] G. Berthelot and Lri-Iie. *Checking properties of nets using transformations*, pages 19–40. Springer Berlin Heidelberg, Berlin, Heidelberg, 1986. (Cited on page 46.)

[20] C. Bors, T. Gschwandtner, S. Kriglstein, S. Miksch, and M. Pohl. Visual interactive creation, customization, and analysis of data quality metrics. *J. Data and Information Quality*, 10(1):3:1–3:26, May 2018. (Cited on page 196.)

[21] J. C. Bose, R. S. Mans, and W. M. P. van der Aalst. Wanna improve process mining results? In *IEEE CIDM 2013*, pages 127–134, 2013. (Cited on pages 174, 175, 176, and 195.)

[22] D. Breuker, M. Matzner, P. Delfmann, and J. Becker. Comprehensible Predictive Models for Business Processes. *MIS Quarterly*, 40(4), 2016. (Cited on page 111.)

[23] S. Buffett and L. Geng. Using classification methods to label tasks in process mining. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(6-7):497–517, 2010. (Cited on page 227.)

[24] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst. A genetic algorithm for discovering process trees. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012. (Cited on pages xxii, 7, 111, 112, 206, and 214.)

[25] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst. A genetic algorithm for discovering process trees. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012. (Cited on pages 210, 211, 213, and 214.)

[26] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst. Quality Dimensions in Process Discovery: The Importance of Fitness, Precision, Generalization and Simplicity. *Int. J. Cooperative Inf. Syst.*, 23(1), 2014. (Cited on page 117.)

[27] Y. Cai and R. de M. Franco. Interactive visualization of network anomalous events. In G. Allen, J. Nabrzyski, E. Seidel, Geert D. van Albada, J. Dongarra, and P. M. A. Sloot, editors, *Computational Science – ICCS 2009*, pages 450–459, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. (Cited on page 196.)

[28] B. C. M. Cappers and J. J. van Wijk. Understanding the context of network traffic alerts. In *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8, Oct 2016. (Cited on page 196.)

[29] B. C. M. Cappers and J. J. van Wijk. Exploring multivariate event sequences using rules, aggregations, and selections. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):532–541, Jan 2018. (Cited on page 144.)

[30] J. Carmona. Projection approaches to process mining using region-based techniques. *Data Mining and Knowledge Discovery*, 24(1):218–246, Jan 2012. (Cited on page 111.)

[31] J. Carmona. *The Alignment of Formal, Structured and Unstructured Process Descriptions*, pages 3–11. Springer International Publishing, Cham, 2017. (Cited on page 142.)

[32] J. Carmona, J. Cortadella, and M. Kishinevsky. Divide-and-conquer strategies for process mining. In Umeshwar Dayal, Johann Eder, Jana Koehler, and Hajo A. Reijers, editors, *Business Process Management*, pages 327–343, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. (Cited on pages 111 and 120.)

[33] J. Carmona, J. Cortadella, and M. Kishinevsky. New region-based algorithms for deriving bounded petri nets. *IEEE Transactions on Computers*, 59(3):371–384, March 2010. (Cited on page 111.)

[34] J. Carmona, B. van Dongen, A. Solti, and M. Weidlich. *Conformance Checking: Relating Processes and Models*. Springer, 2018. (Cited on page 142.)

[35] F. Caron, J. Vanthienen, and B. Baesens. Comprehensive rule-based compliance checking and risk management with process mining. *Decision Support Systems*, 54(3):1357 – 1369, 2013. (Cited on page 144.)

[36] D. Y. Chao and D. T. Wang. Petri net synthesis and synchronization using knitting technique. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 652–657 vol.1, 1994. (Cited on page 48.)

[37] D. Y. Chao, M. C. Zhott, and D. T. Wang. Extending knitting technique to petri net synthesis of automated manufacturing systems. In *Proceedings of the Third International Conference on Computer Integrated Manufacturing,*, pages 56–63, May 1992. (Cited on page 47.)

[38] T. Chatain and J. Carmona. Anti-alignments in Conformance Checking – The Dark Side of Process Models. In F. Kordon and D. Moldt, editors, *Application and Theory of Petri Nets and Concurrency*, pages 240–258, Cham, 2016. Springer International Publishing. (Cited on page 142.)

[39] Y. Cheng, H. Hu, and Y. Liu. Robust supervisor synthesis for automated manufacturing systems using petri nets. In *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1029–1035, Aug 2015. (Cited on page 47.)

[40] F. Chesani, E. Lamma, P. Mello, M. Montali, F. Riguzzi, and S. Storari. *Exploiting Inductive Logic Programming Techniques for Declarative Process Mining*, chapter Transactio, pages 278–295. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. (Cited on pages 111 and 215.)

[41] X. Chu, I. F. Ilyas, and P. Papotti. Holistic data cleaning: Putting violations into context. In *IEEE ICDE*, pages 458–469, apr 2013. (Cited on page 197.)

[42] C. Ciccio and M. Mecella. *Mining Constraints for Artful Processes*, chapter Business I, pages 11–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. (Cited on page 215.)

[43] J. Claes, I. Vanderfeesten, J. Pinggera, H. A. Reijers, B. Weber, and G. Poels. A visual analysis of the process of process modeling. *Information Systems and e-Business Management*, 13(1):147–190, Feb 2015. (Cited on pages 13, 90, 152, and 233.)

[44] R. Conforti, M. Dumas, L. García-Bañuelos, and M. La Rosa. Beyond Tasks and Gateways: Discovering BPMN Models with Subprocesses, Boundary Events and Activity Markers. In Shazia Sadiq, Pnina Soffer, and Hagen Völzer, editors, *Business Process Management*, pages 101–117, Cham, 2014. Springer International Publishing. (Cited on page 111.)

[45] R. Conforti, M. La Rosa, and A. H. M. ter Hofstede. Filtering Out Infrequent Behavior from Business Process Event Logs. *IEEE Trans. Knowl. Data Eng.*, 29(2):300–314, feb 2017. (Cited on pages 195 and 197.)

[46] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri nets from finite transition systems. *IEEE transactions on computers*, 47(8):859–882, 1998. (Cited on pages 111 and 112.)

[47] G. L. Dannenbring. The effect of computer response time on user performance and satisfaction: A preliminary investigation. *Behavior Research Methods & Instrumentation*, 15(2):213–216, Mar 1983. (Cited on page 53.)

[48] T. Dasu. *Data Glitches: Monsters in Your Data*, pages 163–178. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. (Cited on page 196.)

[49] A. Datta and S. Ghosh. Synthesis of a class of deadlock-free Petri nets. *Journal of the ACM (JACM)*, 31(3):486–506, 1984. (Cited on page 47.)

[50] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst. Aligning event logs and declarative process models for conformance checking. In *Business Process Management*, pages 82–97. Springer, 2012. (Cited on pages 142 and 215.)

[51] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst. An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data. *Information Systems*, 47:258–277, 2015. (Cited on page 227.)

[52] M. de Leoni and F. Mannhardt. Road traffic fine management process. (Cited on pages 213 and 214.)

[53] M. de Leoni and A. Marrella. Aligning real process executions and prescriptive process models through automated planning. *Expert Systems with Applications*, 82:162 – 183, 2017. (Cited on page 142.)

[54] A. K. A. de Medeiros, B. F. van Dongen, W. M. P. van der Aalst, and A. J. M. M. Weijters. Process mining for ubiquitous mobile systems: An overview and a concrete algorithm. In L. Baresi, S. Dustdar, H. C. Gall, and M. Matera, editors, *Ubiquitous Mobile Information and Collaboration Systems*, pages 151–165, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. (Cited on page 111.)

[55] A.K. Alves de Medeiros, W.M.P. van der Aalst, and A.J.M.M. Weijters. Quantifying process equivalence based on observed behavior. *Data and Knowledge Engineering*, 64(1):55 – 74, 2008. Fourth International Conference on Business Process Management (BPM 2006) 8th International Conference on Enterprise Information Systems (ICEIS' 2006). (Cited on page 142.)

[56] G. de Vries, R. A. Quintano Neira, G. Geleijnse, P. M. Dixit, and B. F. Mazza. Towards process mining of EMR data - case study for sepsis management. In *Proceedings of the 10th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2017) - Volume 5: HEALTHINF, Porto, Portugal, February 21-23, 2017.*, pages 585–593, 2017. (Cited on page 267.)

[57] J. Desel. Reduction and design of well-behaved concurrent systems. In J C M Baeten and J W Klop, editors, *CONCUR '90 Theories of Concurrency: Unification and Extension*, pages 166–181, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg. (Cited on page 46.)

[58] J. Desel and J. Esparza. *Free choice Petri nets*, volume 40. Cambridge university press, 2005. (Cited on pages 13, 15, 26, 27, 32, 33, 34, 36, 37, 42, 43, 44, 45, 46, 47, 48, 49, 69, 128, and 231.)

[59] P. M. Dixit, J. C. A. M. Buijs, and W. M. P. van der Aalst. Prodigy : Human-in-the-loop process discovery. In *12th International Conference on Research Challenges in Information Science, RCIS 2018, Nantes, France, May 29-31, 2018*, pages 1–12, 2018. (Cited on pages 148 and 266.)

[60] P. M. Dixit, J. C. A. M. Buijs, W. M. P. van der Aalst, B. F. A. Hompes, and J. Buurman. Enhancing process mining results using domain knowledge. In *Proceedings of the 5th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2015), Vienna, Austria, December 9-11, 2015.*, pages 79–94, 2015. (Cited on pages 200 and 266.)

[61] P. M. Dixit, J. C. A. M. Buijs, W. M. P. van der Aalst, B. F. A. Hompes, and J. Buurman. Using domain knowledge to enhance process mining results. In *Data-Driven Process Discovery and Analysis - 5th IFIP WG 2.6 International Symposium, SIMPDA 2015, Vienna, Austria, December 9-11, 2015, Revised Selected Papers*, pages 76–104, 2015. (Cited on pages 200 and 267.)

[62] P. M. Dixit, J. C. A. M. Buijs, H. M. W. Verbeek, and W. M. P. van der Aalst. Fast incremental conformance analysis for interactive process discovery. In *Business Information Systems - 21st International Conference, BIS 2018, Berlin, Germany, July 18-20, 2018, Proceedings*, pages 163–175, 2018. (Cited on pages 116 and 266.)

[63] P. M. Dixit, J. C. A. M. Buijs, H. M. W. Verbeek, and W. M. P. van der Aalst. Fast Incremental Conformance Analysis for Interactive Process Discovery. In Witold Abramowicz and Adrian Paschke, editors, *Business Information Systems*, pages 163–175, Cham, 2018. Springer International Publishing. (Cited on page 143.)

[64] P. M. Dixit, H. S. Garcia Caballero, A. Corvò, B. F. A. Hompes, J. C. A. M. Buijs, and W. M. P. van der Aalst. Enabling interactive process analysis with process mining and visual analytics. In *Proceedings of the 10th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2017) - Volume 5: HEALTHINF, Porto, Portugal, February 21-23, 2017.*, pages 573–584, 2017. (Cited on pages 218 and 266.)

[65] P. M. Dixit, B. F. A. Hompes, N. Tax, and S. J. van Zelst. Handling of Building Permit Applications in The Netherlands: A Multi-Dimensional Analysis. (Cited on page 193.)

[66] P. M. Dixit, S. Suriadi, R. Andrews, M. T. Wynn, A. H. M. ter Hofstede, J. C. A. M. Buijs, and W. M. P. van der Aalst. Detection and interactive repair of event ordering imperfection in process logs. In *Advanced Information Systems Engineering - 30th International Conference, CAiSE 2018, Tallinn, Estonia, June 11-15, 2018, Proceedings*, pages 274–290, 2018. (Cited on pages 172 and 266.)

[67] P. M. Dixit, H. M. W. Verbeek, J. C. A. M. Buijs, and W. M. P. van der Aalst. Interactive data-driven process model construction. In *Conceptual Modeling - 37th International Conference, ER 2018, Xi'an, China, October 22-25, 2018, Proceedings*, pages 251–265, 2018. (Cited on pages 88 and 266.)

[68] P. M. Dixit, H. M. W. Verbeek, and W. M. P. van der Aalst. Fast conformance analysis based on activity log abstraction. In *22nd IEEE International Enterprise Distributed Object Computing Conference, EDOC 2018, Stockholm, Sweden, October 16-19, 2018*, pages 135–144, 2018. (Cited on pages 116 and 266.)

[69] P. M. Dixit, H. M. W. Verbeek, and W. M. P. van der Aalst. Incremental computation of synthesis rules for free-choice petri nets. In *Formal Aspects of Component Software - 15th International Conference, FACS 2018, Pohang, South Korea, October 10-12, 2018, Proceedings*, pages 97–117, 2018. (Cited on pages 52 and 266.)

[70] F. Du, B. Shneiderman, C. Plaisant, S. Malik, and A. Perer. Coping with volume and variety in temporal event sequences: Strategies for sharpening analytic focus. *IEEE Transactions on Visualization and Computer Graphics*, 23(6):1636–1649, June 2017. (Cited on page 196.)

[71] J. Esparza. Synthesis rules for Petri nets, and how they lead to new results. In *International Conference on Concurrency Theory*, pages 182–198. Springer, 1990. (Cited on page 47.)

[72] J. Esparza and P. Hoffmann. Reduction Rules for Colored Workflow Nets. In *Proceedings of the 19th International Conference on Fundamental Approaches to Software Engineering - Volume 9633*, pages 342–358, New York, NY, USA, 2016. Springer-Verlag New York, Inc. (Cited on pages xvi, 46, and 47.)

[73] J. Esparza and M. Silva. On the analysis and synthesis of free-choice systems. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1990*, pages 243–286, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg. (Cited on page 48.)

[74] J. Esparza and M. Silva. Top-down synthesis of live and bounded free-choice nets. In G. Rozenberg, editor, *Advances in Petri Nets 1991*, pages 118–139, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg. (Cited on page 47.)

[75] Javier Esparza, Philipp Hoffmann, and Ratul Saha. Polynomial analysis algorithms for free choice probabilistic workflow nets. *Perform. Eval.*, 117:104–129, 2017. (Cited on page 46.)

[76] D. Fahland, C. Favre, B. Jobstmann, J. Koehler, N. Lohmann, H. Völzer, and K. Wolf. Instantaneous Soundness Checking of Industrial Business Process Models. In U. Dayal, J. Eder, J. Koehler, and H. A Reijers, editors, *Business Process Management*, pages 278–293, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. (Cited on page 46.)

[77] D. Fahland and W. M. P. van der Aalst. *Repairing Process Models to Reflect Reality*, pages 229–245. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. (Cited on page 112.)

[78] D. Fahland and W. M. P. van der Aalst. Repairing process models to reflect reality. In *Business process management*, pages 229–245. Springer, 2012. (Cited on pages 113 and 215.)

[79] M. Fani Sani, S. J. van Zelst, and W. M. P. van der Aalst. Repairing outlier behaviour in event logs. In W. Abramowicz and A. Paschke, editors, *Business Information Systems*, pages 115–131, Cham, 2018. Springer International Publishing. (Cited on page 197.)

[80] N. Fani Sani, S. J. van Zelst, and W. M. P. van der Aalst. Applying sequence mining for outlier detection in process mining. In H. Panetto, C. Debruyne, H. A. Proper, C. A. Ardagna, D. Roman, and R. Meersman, editors, *On the Move to Meaningful Internet Systems. OTM 2018 Conferences*, pages 98–116, Cham, 2018. Springer International Publishing. (Cited on page 197.)

[81] C. Fernandez-Llatas, A. Martinez-Millana, A. Martinez-Romero, J. M. Benedí, and V. Traver. Diabetes care related process modelling using Process Mining techniques. Lessons learned in the application of Interactive Pattern Recognition: coping with the Spaghetti Effect. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2127–2130, aug 2015. (Cited on page 221.)

[82] H. S. Garcia Caballero, A. Corvò, P. M. Dixit, and M. A. Westenberg. Visual analytics for evaluating clinical pathways. In *IEEE Workshop on Visual Analytics in Healthcare, VAHC 2017, Phoenix, AZ, USA, October 1, 2017*, pages 39–46, 2017. (Cited on page 267.)

[83] H. J. Genrich and P. S. Thiagarajan. A theory of bipolar synchronization schemes. *Theoretical Computer Science*, 30(3):241–318, 1984. (Cited on page 46.)

[84] L. Giordano, A. Martelli, M. Spiotta, and D. T. Dupre. Business process verification with constraint temporal answer set programming. *Theory and Practice of Logic Programming*, 13(Special Issue 4-5):641–655, 2013. (Cited on page 215.)

[85] S. Goedertier, D. Martens, B. Baesens, R. Haesen, and J. Vanthienen. Process mining as first-order classification learning on logs with negative events. In *International Conference on Business Process Management*, pages 42–53. Springer, 2007. (Cited on page 227.)

[86] S. Goedertier, D. Martens, J. Vanthienen, and B. Baesens. Robust Process Discovery with Artificial Negative Events. *J. Mach. Learn. Res.*, 10:1305–1340, 2009. (Cited on pages 215 and 216.)

[87] T. J. Goodman and R. Spence. The effect of computer system response time variability on interactive graphical problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(3):207–216, March 1981. (Cited on page 53.)

[88] N. C. Goodwin. Functionality and usability. *Commun. ACM*, 30(3):229–233, March 1987. (Cited on page 53.)

[89] D. Gotz and H. Stavropoulos. Decisionflow: Visual analytics for high-dimensional temporal event sequence data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1783–1792, Dec 2014. (Cited on page 196.)

[90] G. Greco, A. Guzzo, F. Lupa, and P. Luigi. Process Discovery Under Precedence Constraints. *ACM Transactions on Knowledge Discovery from Data*, 9(4):32:1—-32:39, 2015. (Cited on pages 112 and 216.)

[91] G. Greco, A. Guzzo, and L. Pontieri. Process discovery via precedence constraints. In *Proceedings of the 20th European Conference on Artificial Intelligence*, ECAI'12, pages 366–371, Amsterdam, The Netherlands, The Netherlands, 2012. IOS Press. (Cited on page 112.)

[92] T. Gschwandtner, W. Aigner, S. Miksch, J. Gärtner, S. Kriglstein, M. Pohl, and N. Suchy. TimeCleanser: A visual analytics approach for data cleansing of time-oriented data. In *i-KNOW*, page 18. ACM, 2014. (Cited on pages 174 and 197.)

[93] T. Gschwandtner, J. Gärtner, W. Aigner, and S. Miksch. A taxonomy of dirty time-oriented data. *Multidisp. Res. and Prac. for Inf. Sys.*, pages 58–72, 2012. (Cited on pages 174, 175, and 176.)

[94] C. W. Günther and A. Rozinat. Disco: Discover your processes. In *Proceedings of the BPM Demo Session 2012, Co-located with the 10th International Conference on Business Process Management ${$(BPM$}$ 2012), Tallinn, Estonia,*

September 3, 2015, pages 40–44. CEUR Workshop Proceedings, 2012. (Cited on page 219.)

[95] C. W. Günther and W. M. P. van der Aalst. Fuzzy mining – Adaptive process simplification based on multi-perspective metrics. *Business Process Management*, pages 328–343, 2007. (Cited on page 196.)

[96] Q. Guo, L. Wen, J. Wang, Z. Yan, and P. S. Yu. Mining invisible tasks in non-free-choice constructs. In H. R. Motahari-Nezhad, J. Recker, and M. Weidlich, editors, *Business Process Management*, pages 109–125, Cham, 2015. Springer International Publishing. (Cited on page 111.)

[97] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1):10–18, nov 2009. (Cited on page 227.)

[98] M. Hammori, J. Herbst, and N. Kleiner. Interactive workflow mining. In J. Desel, B. Pernici, and M. Weske, editors, *Business Process Management*, pages 211–226, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. (Cited on page 112.)

[99] B. F. A. Hompes, J. C. A. M. Buijs, W. M. P. van der Aalst, P. M. Dixit, and J. Buurman. Detecting change in processes using comparative trace clustering. In *Proceedings of the 5th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2015), Vienna, Austria, December 9-11, 2015.*, pages 95–108, 2015. (Cited on page 267.)

[100] B. F. A. Hompes, J. C. A. M. Buijs, W. M. P. van der Aalst, P. M. Dixit, and J. Buurman. Detecting changes in process behavior using comparative case clustering. In *Data-Driven Process Discovery and Analysis - 5th IFIP WG 2.6 International Symposium, SIMPDA 2015, Vienna, Austria, December 9-11, 2015, Revised Selected Papers*, pages 54–75, 2015. (Cited on page 267.)

[101] P. A. Hsiung and C. Gau. Formal synthesis of real-time embedded software by time-memory scheduling of colored time petri nets11this work was partially supported by research project grant nsc-90-2215-e-194-009 from the national science council, taiwan, roc. *Electronic Notes in Theoretical Computer Science*, 65(6):140 – 159, 2002. Theory and Practice of Timed Systems (Satellite Event of ETAPS 2002). (Cited on page 47.)

[102] M. Der Jeng and F. DiCesare. A review of synthesis techniques for petri nets with applications to automated manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1):301–312, Jan 1993. (Cited on page 47.)

[103] M. O. Kherbouche, N. Laga, and P. A. Masse. Towards a better assessment of event logs quality. In *2016 IEEE SSCI*, pages 1–8, dec 2016. (Cited on pages 174 and 196.)

[104] D. Knuplesch, M. Reichert, L. T. Ly, A. Kumar, and S. Rinderle-Ma. *Visual Modeling of Business Process Compliance Rules with the Support of Multiple Perspectives*, pages 106–120. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. (Cited on page 226.)

[105] S. Kriglstein, M. Pohl, S. Rinderle-Ma, and M. Stallinger. Visual Analytics in Process Mining: Classification of Process Mining Techniques. In Natalia Andrienko and Michael Sedlmair, editors, *EuroVis Workshop on Visual Analytics (EuroVA)*. The Eurographics Association, 2016. (Cited on page 220.)

[106] N. H. Kuiper. Tests concerning random points on a circle. *Indagationes Mathematicae (Proceedings)*, 63:38 – 47, 1960. (Cited on page 180.)

[107] H. Lam, D. Russell, D. Tang, and T. Munzner. Session viewer: Visual exploratory analysis of web session logs. In *2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 147–154, Oct 2007. (Cited on page 144.)

[108] E. Lamma, P. Mello, F. Riguzzi, and S. Storari. *Applying Inductive Logic Programming to Process Mining*, chapter Inductive, pages 132–146. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. (Cited on page 215.)

[109] T. Lammarsch, W. Aigner, A. Bertone, S. Miksch, and A. Rind. Mind the time: Unleashing temporal aspects in pattern discovery. *Computers and Graphics*, 38:38 – 50, 2014. (Cited on page 196.)

[110] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst. Discovering block-structured process models from event logs containing infrequent behaviour. In *Business Process Management Workshops*, pages 66–78. Springer, 2014. (Cited on pages xv, xxii, 9, 10, 89, 111, 112, 137, 192, 194, 196, 212, 213, and 214.)

[111] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst. Process and deviation exploration with inductive visual miner. In *BPM*, 2014. (Cited on pages 220 and 223.)

[112] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst. Scalable process discovery and conformance checking. *Software & Systems Modeling*, 08(Jul):1374–1619, 2016. (Cited on pages xviii, 107, 119, 136, 137, and 143.)

[113] R. Lenz, S. Miksch, M. Peleg, M. Reichert, D. Riaño, and A. ten Teije, editors. *Process Support and Knowledge Representation in Health Care - BPM 2012 Joint Workshop, ProHealth 2012/KR4HC 2012, Tallinn, Estonia, September 3, 2012, Revised Selected Papers*, volume 7738 of *Lecture Notes in Computer Science*. Springer, 2013. (Cited on page 195.)

[114] X. Lu, D. Fahland, and W. M. P. van der Aalst. Conformance checking based on partially ordered event data. In F. Fournier and J. Mendling, editors, *Business Process Management Workshops*, pages 75–88, Cham, 2015. Springer International Publishing. (Cited on page 142.)

[115] L. T. Ly, S. Rinderle-Ma, D. Knuplesch, and P. Dadam. Monitoring business process compliance using compliance rule graphs. In R. Meersman, R. Dillon, P. Herrero, A. Kumar, M. Reichert, L. Qing, B. Ooi, E. Damiani, D. C. Schmidt, J. White, Ma. Hauswirth, P. Hitzler, and M. Mohania, editors, *On the Move to Meaningful Internet Systems: OTM 2011*, pages 82–99, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. (Cited on page 144.)

[116] F. M. Maggi, A. J. Mooij, and W. M. P. van der Aalst. User-guided discovery of declarative process models. In *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on*, pages 192–199. IEEE, 2011. (Cited on pages 112, 207, and 215.)

[117] F. Mannhardt and D. Blinde. Analyzing the Trajectories of Patients with Sepsis using Process Mining. *RADAR+EMISA*, 1859:72–80, 2017. (Cited on pages xxii, 190, 192, and 193.)

[118] F. Mannhardt, M. de Leoni, and H. A. Reijers. The Multi-perspective Process Explorer. In *Proceedings of the BPM Demo Session 2015, Co-located with the 13th International Conference on Business Process Management (BPM 2015), Innsbruck, Austria, September 2, 2015*, pages 130–134. CEUR Workshop Proceedings, 2015. (Cited on page 220.)

[119] R. S. Mans, W. M. P. van der Aalst, and R. J. B. Vanwersch. *Data Quality Issues*, chapter 6, pages 79–88. Springer, 2015. (Cited on pages 175 and 176.)

[120] R. S. Mans, W. M. P. van der Aalst, R. J. B. Vanwersch, and A. J. Moleman. Process mining in healthcare: Data challenges when answering frequently posed questions. In *Process Support and Knowledge Representation in Health Care*, pages 140–153. Springer, 2013. (Cited on page 174.)

[121] K. V. Mardia and P. E. Jupp. *Directional Statistics*. Wiley, 1999. (Cited on page 180.)

[122] B. Mathern, A. Mille, and T. Bellet. An Interactive Method to Discover a Petri Net Model of an Activity. apr 2010. (Cited on page 112.)

[123] J. Mendling, H. A. Reijers, and J. Cardoso. What makes process models understandable? In G. Alonso, P. Dadam, and M. Rosemann, editors, *Business Process Management*, pages 48–63, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. (Cited on page 53.)

[124] J. Mendling, H. A. Reijers, and W. M. P. van der Aalst. Seven process modeling guidelines (7PMG). *Information & Software Technology*, 52(2):127–136, 2010. (Cited on pages 13, 31, 53, 54, and 90.)

[125] J. Mendling, H.A. Reijers, and J. Recker. Activity labeling in process modeling: Empirical insights and recommendations. *Information Systems*, 35(4):467 – 482, 2010. Vocabularies, Ontologies and Rules for Enterprise and Business Process Modeling and Management. (Cited on page 53.)

[126] J. Mendling, H.M.W. Verbeek, B.F. van Dongen, W.M.P. van der Aalst, and G. Neumann. Detection and prediction of errors in epcs of the sap reference model. *Data and Knowledge Engineering*, 64(1):312 – 329, 2008. Fourth International Conference on Business Process Management (BPM 2006) 8th International Conference on Enterprise Information Systems (ICEIS' 2006). (Cited on page 53.)

[127] R. B. Miller. Response Time in Man-computer Conversational Transactions. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), pages 267–277, New York, NY, USA, 1968. ACM. (Cited on pages 46, 53, 54, 233, and 234.)

[128] J. Muñoz-Gama and J. Carmona. A Fresh Look at Precision in Process Conformance. In R. Hull, J. Mendling, and S. Tai, editors, *Business Process Management*, pages 211–226, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. (Cited on pages xviii, 106, 136, and 195.)

[129] J. Muñoz-Gama, J. Carmona, and W. M. P. van der Aalst. Single-Entry Single-Exit decomposed conformance checking. *Information Systems*, 46:102–122, 2014. (Cited on pages 143 and 219.)

[130] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, apr 1989. (Cited on page 4.)

[131] J. Nielsen. Chapter 5 - Usability Heuristics. In *Usability Engineering*, pages 115 – 163. Morgan Kaufmann, San Diego, 1993. (Cited on page 53.)

[132] C. North. Toward measuring visualization insight. *IEEE Computer Graphics and Applications*, 26(3):6–9, 2006. (Cited on page 238.)

[133] J. Pearlman and P. Rheingans. *Visualizing Network Security Events Using Compound Glyphs from a Service-Oriented Perspective*, pages 131–146. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. (Cited on page 196.)

[134] M. Pesic, H. Schonenberg, and W. M. P. van der Aalst. Declare: Full support for loosely-structured processes. In *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, page 287. IEEE, 2007. (Cited on page 207.)

[135] N. Poggi, V. Muthusamy, D. Carrera, and R. Khalaf. Business process mining from e-commerce web logs. In *Business Process Management*, pages 65–80. Springer, 2013. (Cited on page 227.)

[136] E. Ramezani Taghiabadi, D. Fahland, and W. M. P. van der Aalst. Where did I misbehave? diagnostic information in compliance checking. In *Business Process Management*, pages 262–278. Springer Berlin Heidelberg, 2012. (Cited on pages 144 and 215.)

[137] E. Ramezani Taghiabadi, D. Fahland, B. F. van Dongen, and W. M. P. van der Aalst. *Diagnostic Information for Compliance Checking of Temporal Compliance Requirements*, pages 304–320. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. (Cited on page 226.)

[138] H. A. Reijers and J. Mendling. A study into the factors that influence the understandability of business process models. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 41(3):449–462, May 2011. (Cited on page 53.)

[139] A. J. Rembert, A. Omokpo, P. Mazzoleni, and R. T. Goodwin. Process Discovery Using Prior Knowledge. In *Service-Oriented Computing*, pages 328–342. Springer, 2013. (Cited on pages 112 and 215.)

[140] A. Rogge-Solti, R. S. Mans, W. M. P. van der Aalst, and M. Weske. Improving documentation by repairing event logs. In *Proc. PoEM 2013*, pages 129–144. Springer, 2013. (Cited on pages 175, 193, and 197.)

[141] A. Rogge-Solti, R. S. Mans, W. M. P. van der Aalst, and M. Weske. Repairing event logs using timed process models. In *OTM*, pages 705–708. Springer, 2013. (Cited on pages 175 and 197.)

[142] A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008. (Cited on page 142.)

[143] A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008. (Cited on page 143.)

[144] W. Runte and M. El Kharbili. Constraint Checking for Business Process Management. In *GI Jahrestagung*, pages 4093–4103, 2009. (Cited on page 215.)

[145] H. Schulz, T. Nocke, M. Heitzler, and H. Schumann. A Design Space of Visualization Tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2366–2375, dec 2013. (Cited on page 220.)

[146] M. Sgroi, L. Lavagno, Y. Watanabe, and A. Sangiovanni-Vincentelli. Synthesis of embedded software using free-choice petri nets. In *Proceedings of the 36th Annual ACM/IEEE Design Automation Conference*, DAC '99, pages 805–810, New York, NY, USA, 1999. ACM. (Cited on page 47.)

[147] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1997. (Cited on page 53.)

[148] M. Solé and J. Carmona. *Incremental Process Discovery*, pages 221–242. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. (Cited on page 111.)

[149] S. Song, Y. Cao, and J. Wang. Cleaning Timestamps with Temporal Constraints. *Proc. VLDB Endow.*, 9(10):708–719, 2016. (Cited on page 197.)

[150] W. Song, X. Xia, H. A. Jacobsen, P. Zhang, and H. Hu. Heuristic Recovery of Missing Events in Process Logs. In *ICWS*, pages 105–112, 2015. (Cited on page 197.)

[151] S. Suriadi, R. Andrews, A. H. M. ter Hofstede, and M. T. Wynn. Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. *Information Systems*, 64:132–150, 2017. (Cited on pages 174, 175, 176, 195, and 236.)

[152] I. Suzuki and T. Murata. A method for stepwise refinement and abstraction of petri nets. *Journal of Computer and System Sciences*, 27(1):51 – 76, 1983. (Cited on page 47.)

[153] M. Swennen, G. Janssenswillen, M. Jans, B. Depaire, and K. Vanhoof. Capturing Process Behavior with Log-Based Process Metrics. In *SIMPDA*, 2015. (Cited on pages 174 and 196.)

[154] F. Taymouri and J. Carmona. *A Recursive Paradigm for Aligning Observed Behavior of Large Structured Process Models*, pages 197–214. Springer International Publishing, Cham, 2016. (Cited on page 144.)

[155] F. Taymouri and J. Carmona. Model and Event Log Reductions to Boost the Computation of Alignments. *6th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2016)*, 2016. (Cited on page 143.)

[156] R. Valette. Analysis of Petri nets by stepwise refinements. *Journal of Computer and System Sciences*, 18(1):35 – 46, 1979. (Cited on page 46.)

[157] S. K. L. M. van den Broucke and J. De Weerdt. Fodina: A robust and flexible heuristic process discovery technique. *Decision Support Systems*, 100:109 – 118, 2017. Smart Business Process Management. (Cited on page 111.)

[158] W. M. P. van der Aalst. The application of Petri nets to workflow management. *Journal of circuits, systems, and computers*, 8(01):21–66, 1998. (Cited on pages 25 and 26.)

[159] W. M. P. van der Aalst. *Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques*, pages 161–183. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000. (Cited on page 27.)

[160] W. M. P. van der Aalst. On the Representational Bias in Process Mining. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2011 20th IEEE International Workshops on*, pages 2–7, 2011. (Cited on page 89.)

[161] W. M. P. van der Aalst. Using process mining to bridge the gap between bi and bpm. *Computer*, 44(12):77–80, Dec 2011. (Cited on pages xv and 7.)

[162] W. M. P. van der Aalst. Decomposing Petri nets for process mining: A generic approach. *Distributed and Parallel Databases*, 31(4):471–507, dec 2013. (Cited on pages 19, 20, 119, 120, and 143.)

[163] W M P van der Aalst. *Process Mining - Data Science in Action, Second Edition*. Springer, 2016. (Cited on pages 3, 20, 118, 173, 175, 176, 195, 210, and 236.)

[164] W. M. P. van der Aalst and A. H. M. de Leoni, M.and ter Hofstede. Process mining and visual analytics: Breathing life into business process models. 2011. (Cited on page 220.)

[165] W. M. P. van der Aalst and Others. Process Mining Manifesto. In *BPM 2011 Workshops (1)*, pages 169–194. Springer-Verlag, 2011. (Cited on pages 174 and 175.)

[166] W. M. P. van der Aalst, V. Rubin, H. M. W. Verbeek, B. F. van Dongen, E. Kindler, and C. W. Günther. Process mining: a two-step approach to balance between underfitting and overfitting. *Software & Systems Modeling*, 9(1):87, Nov 2008. (Cited on page 111.)

[167] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, Jul 2003. (Cited on pages 13 and 31.)

[168] W. M. P. van der Aalst, K. M. van Hee, A. H. M. ter Hofstede, N. Sidorova, H. M. W. Verbeek, M. Voorhoeve, and M. T. Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects of Computing*, 23(3):333–363, 2011. (Cited on pages 26, 112, and 231.)

[169] W. M. P. van der Aalst, A. J. M. M. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *Knowledge and Data Engineering, IEEE Transactions on*, 16(9):1128–1142, 2004. (Cited on pages 89, 110, and 112.)

[170] J. M. E. M. van der Werf, B. F. van Dongen, C. A. J. Hurkens, and A. Serebrenik. Process discovery using integer linear programming. In *International Conference on Applications and Theory of Petri Nets*, pages 368–387. Springer, 2008. (Cited on pages 106, 111, 112, 137, and 214.)

[171] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst. The ProM framework: A new era in process mining tool support. In *Application and Theory of Petri Nets (ICATPN)*, volume 3536, pages 444–454, 2005. (Cited on pages 101, 105, 136, 161, 188, 211, and 221.)

[172] B. F. van Dongen, W. M. P. van der Aalst, and H. M. W. Verbeek. *Verification of EPCs: Using Reduction Rules and Petri Nets*, pages 372–386. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. (Cited on page 46.)

[173] S. K. L. M. vanden Broucke, J. Munoz-Gama, J. Carmona, B. Baesens, and J. Vanthienen. *Event-Based Real-Time Decomposed Conformance Analysis*, pages 345–363. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. (Cited on page 144.)

[174] S. K. L. M. vanden Broucke, J. De Weerdt, J. Vanthienen, and B. Baesens. Determining process model precision and generalization with weighted artificial negative events. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1877–1889, Aug 2014. (Cited on page 142.)

[175] I. Vanderfeesten, H. A. Reijers, J. Mendling, W. M. P. van der Aalst, and J. Cardoso. On a quest for good process models: The cross-connectivity metric. In Z. Bellahsène and M. Léonard, editors, *Advanced Information Systems Engineering*, pages 480–494, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. (Cited on page 53.)

[176] J. Vanhatalo, H. Völzer, and F. Leymann. Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition. In B. J. Krämer, K. Lin, and P. Narasimhan, editors, *Service-Oriented Computing – ICSOC 2007*, pages 43–55, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. (Cited on page 46.)

[177] B. Vázquez-Barreiros, S. J. van Zelst, J. C. A. M. Buijs, M. Lama, and M. Mucientes. *Repairing Alignments: Striking the Right Nerve*, pages 266–281. Springer International Publishing, Cham, 2016. (Cited on page 143.)

[178] H. M. W. Verbeek. *Decomposed Replay Using Hiding and Reduction as Abstraction*, pages 166–186. Springer Berlin Heidelberg, Berlin, Heidelberg, 2017. (Cited on pages 136 and 143.)

[179] H. M. W. Verbeek and W. M. P. van der Aalst. *Woflan 2.0 A Petri-Net-Based Workflow Diagnosis Tool*, pages 475–484. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000. (Cited on page 46.)

[180] H. M. W. Verbeek, W. M. P. van der Aalst, and J. Munoz-Gama. Divide and Conquer: A Tool Framework for Supporting Decomposed Discovery in Process Mining. *The Computer Journal*, 60(11):1649–1674, 2017. (Cited on pages 136 and 143.)

[181] S. Wagner, M. W. Beckmann, B. Wullich, C. Seggewies, M. Ries, T. Bürkle, and H. Prokosch. Analysis and classification of oncology activities on the way to workflow based single source documentation in clinical information systems. *BMC Medical Informatics and Decision Making*, 15(1):107, dec 2015. (Cited on page 161.)

[182] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens. A robust f-measure for evaluating discovered process models. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 148–155, April 2011. (Cited on page 142.)

[183] M. Weidlich, A. Polyvyanyy, N. Desai, and J. Mendling. Process compliance measurement based on behavioural profiles. In B. Pernici, editor, *Advanced Information Systems Engineering*, pages 499–514, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. (Cited on page 143.)

[184] M. Weidlich, A. Polyvyanyy, N. Desai, J. Mendling, and M. Weske. Process compliance analysis based on behavioural profiles. *Information Systems*, 36(7):1009 – 1025, 2011. Special Issue: Advanced Information Systems Engineering (CAiSE'10). (Cited on page 143.)

[185] A. J. M. M. Weijters and J. T. S. Ribeiro. Flexible heuristics miner (fhm). In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 310–317, April 2011. (Cited on pages 89, 106, 111, 112, 179, 194, and 196.)

[186] A. J. M. M. Weijters, W. M. P. van der Aalst, and A. K. A. de Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166:1–34, 2006. (Cited on page 111.)

[187] L. Wen, W. M. P. van der Aalst, J. Wang, and J. Sun. Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, 15(2):145–180, Oct 2007. (Cited on page 111.)

[188] L. Wen, J. Wang, and J. Sun. Detecting implicit dependencies between tasks from event logs. In X. Zhou, J. Li, H. T. Shen, M. Kitsuregawa, and Y. Zhang, editors, *Frontiers of WWW Research and Development - APWeb 2006*, pages 591–603, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. (Cited on page 111.)

[189] L. Wen, J. Wang, and J. Sun. Mining invisible tasks from event logs. In G. Dong, X. Lin, W. Wang, Y. Yang, and J. X. Yu, editors, *Advances in Data and Web Management*, pages 358–365, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. (Cited on page 111.)

[190] L. Wen, J. Wang, W. M.P. van der Aalst, B. Huang, and J. Sun. Mining process models with prime invisible tasks. *Data and Knowledge Engineering*, 69(10):999 – 1021, 2010. (Cited on page 111.)

[191] J. M. E. M. Werf, B. F. van Dongen, C. A. J. Hurkens, and A. Serebrenik. *Process Discovery Using Integer Linear Programming*, chapter Applicatio, pages 368–387. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. (Cited on pages 215 and 216.)

[192] K. Wolf. Generating Petri Net State Spaces. In Jetty Kleijn and Alex Yakovlev, editors, *Petri Nets and Other Models of Concurrency – ICATPN 2007*, pages 29–42, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. (Cited on page 46.)

[193] K. Wongsuphasawat and D. Gotz. Exploring flow, factors, and outcomes of temporal event sequences with the outflow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2659–2668, Dec 2012. (Cited on page 196.)

[194] E. Zgraggen, S. Drucker, D. Fisher, R. DeLine, and R. DeLine. (s|qu)eries: Visual regular expressions for querying and exploring event sequences. ACM - Association for Computing Machinery, April 2015. Proceedings of CHI 2015. (Cited on page 144.)

Summary

Interactive Process Mining

A process is a series of actions performed in order to achieve a particular task or goal. Processes are omnipresent, for example, a manufacturing process which deals with the production of some goods, a loan application process in a bank, a hospital visit of a patient. The human understanding of these processes is usually facilitated by representing them as intuitive graphical models. Typically a domain expert, who has a high level overview of the end-to-end execution of a process, constructs the expected process model. Since these modeling notations offer intuitive visualization of processes, the resulting models have proved to be valuable artifacts to enable communication of complex knowledge in a comprehensible way across people from dissimilar backgrounds. The process model as described by a domain expert is the best guess, of how a process acts, or should act. However, reality may not always conform to this expected behavior of the process.

The execution histories of processes, called *event logs*, can be easily extracted from the corresponding information systems. These event logs provide a view on the execution of processes in reality and can thus be used to discover process models when the process model is unknown, or to analyze the behavior of the process in reality. This branch of science which uses event logs to analyze the process behavior is known as *process mining*.

Process modeling and process mining are typically on the extreme ends of a spectrum. On one end, we have the process mining techniques, which analyze event logs in order to *automatically* deduce insights, for example discovering process models. The user has very little influence when using such techniques, and usually, it is not possible to incorporate the domain knowledge during process mining. On the other end, we have the traditional process modeling tools which are completely user driven and use no historical evidence from the event logs during process modeling. In the manual process modeling scenario, the user may miss out on important real-life execution information that may be present in the event logs. Whereas the automated process mining techniques may produce results which may be inaccurate and incomprehensible due to noisy and/or incomplete event logs.

In this thesis, we addressed this gap between the traditional *human-driven* process modeling world, with the *data-driven* process mining world, by developing techniques that enable interactive process mining. We identified and addressed four research

goals:

- *Interactive event log repair*: Addressed the issue of repairing the source of process mining analysis, i.e., the event log, with available domain knowledge of an informed user.
- *Interactive process model construction*: Addressed the issue of modeling/discovering process models interactively by using both information derived from the event log and the expertise of a user. Techniques were developed to assist the user in decision-making by
  - providing fast conformance analysis between the event log and the process being modeled.
  - providing recommendations to enable semi-automated process discovery by combining automated and interactive approaches for process modeling.

  Furthermore, we provided formal guarantees for the resulting process model. To ensure the soundness of the constructed process model, we used liveness and boundedness preserving synthesis rules for free-choice Petri nets.
- *Interactive process model repair*: Addressed the issue of interactively repairing a pre-existing process model by using the domain knowledge and information derived from the event log.
- *Interactive process analytics*: Addressed the issue of interactively exploring real-life execution of a process based on the process model and the event log, in order to evaluate the possible compliance and performance oriented issues.

The proposed techniques were implemented and evaluated using several real-life scenarios with real event data and/or process experts.

Acknowledgments

First of all, I would like to thank my first promoter, Wil van der Aalst. He has given me immense support and valuable scientific advise throughout the four years. I would also like to convey a special thanks for the reassurances and guidance whenever I had doubts about my PhD at various points. Working with Wil has helped me to grow both scientifically and personally. Besides Wil, I was lucky to have multiple supervisors. I would like to thank Joos, for his help and guidance during the first two years of my PhD. Having a friendly supervisor like Joos, who always had time to have a chat, made settling into a new setting very comfortable and easy. Additionally, I would like to thank Eric for taking over the supervision duties from Joos, during the latter years of my PhD. Many thanks especially for assisting me in the formal aspects during my PhD. Looking back, I can say that I have enjoyed working on the formal side of things (even though at the time, it seemed never-ending).

Due to the flagship collaboration with Philips, I also had the pleasure to work in an industrial research setting. I would like to convey my thanks to Hans Buurman for all his help and support during the initial years of my PhD. I enjoyed my discussions with him, and appreciate the efforts put in by him to provide exposure to data analysis in healthcare. I would also like to thank Jan Korst for taking over the supervision from Philips side, half-way through my PhD. He is one of the nicest persons I know, and it was a pleasure to work with him.

A special thanks to my colleagues and friends from Optimizing Healthcare Workflows stream, Bart, Humberto and Alberto for all the nice collaborations and for sharing this PhD journey with me. A special thanks to Niek and Alberto for accepting to be my seconds. Also, I would like to thank Jack, Michel and Mark for all the nice inputs and discussions during the monthly meetings. I would also like to thank the committee members for reviewing my thesis and for their valuable feedback.

During my PhD, I had several interesting collaboration opportunities. I would like to thank Arthur, Moe, Robert and Suriadi for the constructive collaboration during my visit to the BPM group at Queensland University of Technology, Brisbane. Furthermore, thanks to all my colleagues at Philips for a wonderful atmosphere and good collaborations.

Besides the scientific aspects of doing a PhD, I would like to thank the secretaries of our group for having answers to almost all of my questions. A special thanks to Ine, for proof-reading this thesis, despite her busy schedule. Also, thanks to Odette for handling all my administrative work at Philips.

Doing a PhD can be a lonesome experience. A part of it was addressed when I got married mid-way into my PhD. Nrupa, thanks for everything! I am also extremely thankful to all the past and present colleagues and friends from the process mining group at TU/e, my friends from flagship collaboration, and all the other amazing friends that I made in Eindhoven, for making these last four years enjoyable. A special thanks to my parents, brother, sister-in-law and nephew for their unconditional love and support.

P.M. (Alok) Dixit
Eindhoven, June 2019

Curriculum Vitae

Prabhakar (Alok) Madhukar Dixit was born on 05-10-1986 in Ponda, Goa, India. After completing Bachelors of Technology in Computer Science & Engineering in 2008 at National Institute of Technology in Durgapur, India, he worked as a software developer at Computer Sciences Corporation in India. Then, he pursued a joint MSc course (International Master in Service Engineering) offered by University of Stuttgart, in Germany, University of Crete, in Greece and Tilburg University in The Netherlands. From February 2014, he started a PhD project at Eindhoven University of Technology, The Netherlands in collaboration with Philips research, of which the results are presented in this dissertation. Since March 2019, he is employed at KPMG, Eindhoven.

List of Publications

Prabhakar (Alok) M. Dixit has the following publications:

Relevant Proceedings

- P. M. Dixit, J. C. A. M. Buijs, H. M. W. Verbeek, and W. M. P. van der Aalst. Fast incremental conformance analysis for interactive process discovery. In *Business Information Systems - 21st International Conference, BIS 2018, Berlin, Germany, July 18-20, 2018, Proceedings*, pages 163–175, 2018

- P. M. Dixit, S. Suriadi, R. Andrews, M. T. Wynn, A. H. M. ter Hofstede, J. C. A. M. Buijs, and W. M. P. van der Aalst. Detection and interactive repair of event ordering imperfection in process logs. In *Advanced Information Systems Engineering - 30th International Conference, CAiSE 2018, Tallinn, Estonia, June 11-15, 2018, Proceedings*, pages 274–290, 2018

- P. M. Dixit, H. M. W. Verbeek, and W. M. P. van der Aalst. Fast conformance analysis based on activity log abstraction. In *22nd IEEE International Enterprise Distributed Object Computing Conference, EDOC 2018, Stockholm, Sweden, October 16-19, 2018*, pages 135–144, 2018

- P. M. Dixit, H. M. W. Verbeek, J. C. A. M. Buijs, and W. M. P. van der Aalst. Interactive data-driven process model construction. In *Conceptual Modeling - 37th International Conference, ER 2018, Xi'an, China, October 22-25, 2018, Proceedings*, pages 251–265, 2018

- P. M. Dixit, H. M. W. Verbeek, and W. M. P. van der Aalst. Incremental computation of synthesis rules for free-choice petri nets. In *Formal Aspects of Component Software - 15th International Conference, FACS 2018, Pohang, South Korea, October 10-12, 2018, Proceedings*, pages 97–117, 2018

- P. M. Dixit, J. C. A. M. Buijs, and W. M. P. van der Aalst. Prodigy : Human-in-the-loop process discovery. In *12th International Conference on Research Challenges in Information Science, RCIS 2018, Nantes, France, May 29-31, 2018*, pages 1–12, 2018

- P. M. Dixit, H. S. Garcia Caballero, A. Corvò, B. F. A. Hompes, J. C. A. M. Buijs, and W. M. P. van der Aalst. Enabling interactive process analysis with process mining and visual analytics. In *Proceedings of the 10th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2017) - Volume 5: HEALTHINF, Porto, Portugal, February 21-23, 2017.*, pages 573–584, 2017

- P. M. Dixit, J. C. A. M. Buijs, W. M. P. van der Aalst, B. F. A. Hompes, and J. Buurman. Enhancing process mining results using domain knowledge. In *Proceedings of the 5th International Symposium on Data-driven Process Discovery*

and Analysis (SIMPDA 2015), Vienna, Austria, December 9-11, 2015., pages 79–94, 2015

- P. M. Dixit, J. C. A. M. Buijs, W. M. P. van der Aalst, B. F. A. Hompes, and J. Buurman. Using domain knowledge to enhance process mining results. In *Data-Driven Process Discovery and Analysis - 5th IFIP WG 2.6 International Symposium, SIMPDA 2015, Vienna, Austria, December 9-11, 2015, Revised Selected Papers*, pages 76–104, 2015

Other Proceedings

- G. de Vries, R. A. Quintano Neira, G. Geleijnse, P. M. Dixit, and B. F. Mazza. Towards process mining of EMR data - case study for sepsis management. In *Proceedings of the 10th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2017) - Volume 5: HEALTHINF, Porto, Portugal, February 21-23, 2017.*, pages 585–593, 2017

- H. S. Garcia Caballero, A. Corvò, P. M. Dixit, and M. A. Westenberg. Visual analytics for evaluating clinical pathways. In *IEEE Workshop on Visual Analytics in Healthcare, VAHC 2017, Phoenix, AZ, USA, October 1, 2017*, pages 39–46, 2017

- B. F. A. Hompes, J. C. A. M. Buijs, W. M. P. van der Aalst, P. M. Dixit, and J. Buurman. Detecting changes in process behavior using comparative case clustering. In *Data-Driven Process Discovery and Analysis - 5th IFIP WG 2.6 International Symposium, SIMPDA 2015, Vienna, Austria, December 9-11, 2015, Revised Selected Papers*, pages 54–75, 2015

- B. F. A. Hompes, J. C. A. M. Buijs, W. M. P. van der Aalst, P. M. Dixit, and J. Buurman. Detecting change in processes using comparative trace clustering. In *Proceedings of the 5th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2015), Vienna, Austria, December 9-11, 2015.*, pages 95–108, 2015