UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

COMPUTER SCIENCE DEPARTMENT

Master of Science in Computer Science

Formal methods for Computer Science

Adaptive Petri Net for the Well-being of the Elderly

Link GitHub:
https://github.com/VitoNicolaLosavio/ProcessMiningProject.git

Group:

Nicolas Pinto 807348
Vito Nicola Losavio 807222

A.Y. 2023/2024

Contents

1 Introduction and Context **4**

2 Data Collection and Data Generation **6**
2.1 Introduction 6
2.2 Original Dataset 6
2.2.1 Activity Categories 6
2.2.2 Dataset Structure 7
2.3 Extension of Dataset through ChatGPT 7
2.3.1 Extension Procedure 7
2.4 Synthetic Dataset 8
2.4.1 Activity Categories 8
2.4.2 Dataset Structure 9
2.4.3 Results Analysis 9

3 Process-Mining Tools **12**
3.1 PM4PY 12
3.2 Heuristic Miner 12
3.3 Inductive Miner 13

4 Process Model Creation and Use Case Description **15**
4.1 Process Model Creation 15
4.1.1 Comparison between Petri Net 16
4.2 Use Case Description 17
4.2.1 Experiment 1: Evaluation of Pre-Trained Petri Net 18
4.2.2 Experiment 2: Detection and Management of Unknown Activities 18
4.2.3 Experiment 3: Petri Net Generation and Evaluation 18
4.2.4 Conclusion 18

5 Results and Conclusions **19**
5.1 Results Analysis 19
5.2 Conclusions 21
5.3 Future Enhancements 21
5.3.1 Design Enhancements 21

5.3.2 Implementation Enhancements 22

Bibliography **23**

Chapter 1

Introduction and Context

Smart homes, characterized by the advanced integration of information and communications technologies into everyday devices, are rapidly becoming a crucial part of modern societies. The interconnection of devices within a home environment offers a wide range of opportunities, from efficient energy management to the automation of daily tasks, helping to improve quality of life and optimize resource use. In this context, the analysis of the activities carried out within a smart home plays a fundamental role to understand the behavioral patterns of users, identify inefficiencies and optimize the user experience.

The primary objective of this project is to extract behavioral patterns through the application of Process Mining techniques within the context of a smart nursing home. These patterns will serve as a foundation for the detection of anomalous actions performed by elderly residents. The focus is twofold: first, to identify patterns that represent intentional deviations from regular behavior, and second, to differentiate between intentional anomalies and potentially unintended actions. In the event that an anomalous action is deemed intentional, it is incorporated into the set of possible actions. Conversely, if an identified anomaly is determined to be unintentional, the system intervenes to provide assistance to the elderly resident.

By leveraging Process Mining, the project aims to comprehensively analyze the rich dataset capturing the daily activities of elderly residents. The extracted patterns will act as a baseline representing normal behavioral sequences. Deviations from these established patterns will be flagged and scrutinized for potential anomalies. The intention is to discern intentional variations, indicative of personal preferences or adaptations, from non-intentional anomalies that may signal a need for intervention.

The proactive nature of the system lies in its ability to not only identify anomalies but also respond accordingly. If an anomalous action is intentional, the system adapts to recognize it as part of the resident's behavioral repertoire. Conversely, if an anomaly is identified as unintentional, the system initiates assistance measures, ensuring the well-being and safety of the elderly resident.

This goal is achieved by comparing a dataset obtained through the log of the installed devices and an artificially extended version of this dataset. In the next

section, the context of the project, the nature of the datasets used and the specific objectives to be achieved will be presented in detail.

Finally, the documentation will conclude with a critical evaluation of the results obtained and with the discussion of the potential practical applications of the discoveries emerged from the analysis of processes in a smart home.

Chapter 2

Data Collection and Data Generation

2.1 Introduction

In this chapter, we will describe the original dataset used as a starting point for our project and explain how we extended and enriched this dataset using ChatGPT.

2.2 Original Dataset

The original dataset used for this project (Figure 2.1) was collected from reliable sources within the context of smart nursing home. It includes records of activities performed by many elderly individuals within a care facility.

2.2.1 Activity Categories

The activities recorded in the original dataset include:

- brush_hair
- sit_down
- free_activity
- eat_meal
- drink
- play_phone
- read

2.2.2 Dataset Structure

The original dataset is structured with the following columns:

- Event_Timestamp: Integer number that indicates the order in which the action is performed relative to the others.
- Type_Event: Type of action performed (e.g., 'begin_process', 'end_process', 'begin_activity', 'end_activity', 'context_description').
- Reference_Workflow: Day of the week when the action was performed.
- Case_ID: Case identifier.
- Name_Activity: Name of the performed activity.
- Occurrency_Activity: Progressive number of occurrences of 'Name_Activity'.

```
entry(1,begin_of_process,monday,monday6637,none,none).
entry(2,begin_of_activity,monday,monday6637,sit_down,1).
entry(3,begin_of_activity,monday,monday6637,eat_meal,1).
entry(4,end_of_activity,monday,monday6637,eat_meal,1).
entry(5,end_of_activity,monday,monday6637,sit_down,1).
entry(8,begin_of_activity,monday,monday6637,read,1).
```

Figure 2.1: A portion of the original dataset.

2.3 Extension of Dataset through ChatGPT

To obtain an extended and more diversified version of the original dataset, we leveraged the OpenAI ChatGPT language model. We generated simulated interactions of elderly individuals in a smart nursing home, covering a wide range of scenarios and activities easily identified by computer vision or speech recognition techniques. The approach involved interacting with the model, asking it to describe specific actions or situations in which the elderly individuals might find themselves.

2.3.1 Extension Procedure

The procedure followed to extend the dataset was as follows:

1. Interaction with ChatGPT: We interacted with ChatGPT by presenting specific scenarios related to elderly activities. 2.4.3
2. Generation of Simulated Text: ChatGPT generated simulated text in response to our interactions, creating detailed and realistic scenarios.
3. Integration into the Dataset: The newly generated data was integrated into the original dataset, maintaining the same structure and activity categories.

2.4 Synthetic Dataset

The new dataset (Figure 2.2), synthetically generated through the use of ChatGPT captures various activities performed by seniors, both regular and extraordinary, including but not limited to brushing hair, eating meals, engaging in physical activity, and carrying out special activities. These are actions that can be performed by the elderly and can be easily detected by well-known technologies.

The rules, always contained in the prompt at the end of the chapter, ensure that the generated examples align with realistic sequences of actions, considering the constraints imposed by daily life. According to such rules:

- Actions can be performed in any order;
- Actions can be performed simultaneously;
- The sequence of performed actions should be consistent and realistic. Seniors cannot perform conflicting activities simultaneously (e.g., eating and sleeping);
- Extraordinary activities are allowed during public holidays;
- Seniors cannot go to the bathroom simultaneously.

2.4.1 Activity Categories

The dataset simulates the daily activities of seniors residing in a smart home environment, focusing on activities that contribute to their well-being and independence. The dataset includes various types of events, such as the start and end of processes, activities, and context descriptions. Each senior is identified by a unique Senior_ID, and activities are timestamped for accurate representation of daily routines.

This dataset, in addition to the actions already present in the original dataset, includes the following new actions:

- carry_out_special_activities
- receive_visits
- be_in_bathroom
- be_in_company
- hobby
- physical_activity
- sleep
- take_medicines

2.4.2 Dataset Structure

The extended dataset is structured with the following columns:

- Event_Timestamp: Returns the date and time the activity was recorded.
- Type_Event: Type of action performed (e.g., 'begin_process', 'end_process', 'begin_activity', 'end_activity', 'context_description').
- Reference_Workflow: Day of the week when the action was performed.
- Case_ID: Case identifier.
- Name_Activity: Name of the performed activity.
- Occurrency_Activity: Progressive number of occurrences of 'Name_Activity'.
- Senior_ID: Identifier of the Senior.

```
Monday_S1,01-01-24 08:00:00,start_process,Monday,S1,none,none
Monday_S1,01-01-24 08:07:12,begin_activity,Monday,S1,sit_down,1
Monday_S2,01-01-24 08:00:00,start_process,Monday,S2,none,none
Monday_S2,01-01-24 08:08:30,begin_activity,Monday,S2,physical_activity,1
Monday_S3,01-01-24 08:00:00,start_process,Monday,S3,none,none
Monday_S3,01-01-24 08:10:45,begin_activity,Monday,S3,eat_meal,1
Monday_S1,01-01-24 08:15:25,end_activity,Monday,S1,sit_down,1
```

Figure 2.2: A portion of the extended dataset.

A major change from the original dataset was the event-timestamp. By having timestamps associated with each activity, one can start to reconstruct the sequence of events. This temporal sequencing is essential for several reasons. Firstly, it allows us to analyze the duration of each activity or the process as a whole. This helps in pinpointing bottlenecks or areas where efficiency can be improved. Secondly, it enables us to trace the path followed by a process instance over time. This can reveal variations in process flows and highlight deviations from the norm.

2.4.3 Results Analysis

Extending the dataset through ChatGPT led to a significant increase in the diversity of simulated activities and contributed to improving the model's generalization, enhancing the predictive capability of the model regarding elderly activities in a smart nursing home.

This concludes the chapter on data collection and preprocessing for our project. In the next chapters, we will examine the model architecture used and details of the training process.

Prompt used with ChatGPT

```
Generates a table in which are reported the actions performed by a
group of seniors within a smart home that applies computer
vision are specified. The table, must contain the following
columns: ⟨Case_ID, Event_Time_Stamp, Type_Of_Event,
Reference_Workflow, Case_ID, Name_Activity,
Occurrency_Of_Activity⟩, where Case_ID is the senior identifier,
Event_Time_Stamp is the date and time when the action is
performed (in the format mm-dd-yy hh-mm-ss), Type_Of_Event is the
type of the performed action (one of 'begin_process',
'end_process', 'begin_activity', 'end_activity',
'context_description'), Reference_Workflow is the name of the day
in the week in which the action is performed (after 50 actions
for each senior you move on to the next day), Case_ID formed by
reference_Workflow+Case_ID , Name_Activity is the name of the
performed activity, and Occurrency_Of_Activity is the progressive
number of occurrence of Name_Activity.

The actions regularly allowed during the week are:

Brush_Hair
Drink
Eat_Meal
Play_Phone
Read
Sit_Down
Be_In_Bathroom
Take_Medicines
Physical_Activity
Sleep
Be_In_Company
Hobby

The extraordinary actions allowed during public holidays are:

Receive_Visits
Carry_Out_Special_Activities

RULES: Actions can be carried out in any order, but make sure
that the sequence of actions taken is consistent and akin to
reality. For example, check that a person cannot sleep and
perform other actions at the same time, or, a senior does not
begin exercise before leaving the bathroom. Generate examples
```

```
taking into account that multiple activities can be carried out
simultaneously,  for example a person can eat while sitting. Keep
in mind that not all activities can be carried out at the same
time, for example a person cannot eat and exercise at the same time.

OUTPUT: The output must be a DataFrame.
```

Chapter 3

Process-Mining Tools

3.1 PM4PY

PM4Py [2], which stands for Process Mining for Python, is an open-source library designed to facilitate the analysis of processes through the use of process mining techniques. Process mining is a discipline that involves extracting information, models, and knowledge from event logs recorded by information systems during the execution of business processes.

It is provides tools and algorithms to perform various process mining activities, such as process discovery, conformance checking, performance analysis, and process model visualization. The library is developed in Python and offers a wide range of functionalities that allow users to analyze process data and gain a deeper understanding of how activities are carried out within an organization.

Such library is used in various industries, including business process management (BPM), logistics, healthcare, and other contexts where understanding and optimizing business workflows are crucial.

3.2 Heuristic Miner

The Heuristic Miner [1] is a process mining algorithm designed to automatically construct a process model from event logs. It employs heuristics to guide the creation of a model based on observed sequences of activities in the event log. The main goal of the Heuristic Miner is to provide a simple and efficient method for generating a process model that represents the most likely order of activities in a given process.

Here are the key characteristics of the Heuristic Miner algorithm:

- Heuristics: The algorithm uses heuristics to guide the construction of the process model. Heuristics are rules or strategies that help make decisions based on incomplete or uncertain information.
- Frequency-based: The Heuristic Miner often relies on the frequency of observed activity sequences in the event log. Activities that frequently occur together

are likely to be connected in the resulting process model.

- Parallelism handling: The algorithm can handle parallelism in the process by identifying activities that often occur concurrently. This allows the generated process model to represent parallel branches in the process flow.

- Loop handling: The Heuristic Miner is capable of handling loops or repetitive patterns in the process. It identifies activities that are often repeated in the event log and represents them as loops in the resulting process model.

- Noise handling: The algorithm is designed to handle noisy event logs where there might be variations or deviations in the observed sequences of activities.

It's important to note that while the Heuristic Miner is a powerful and commonly used algorithm, it may not always produce an optimal or precise process model. Since the generated model is based on heuristics and statistical information from the event log, and the accuracy depends on the quality and characteristics of the input data.

3.3 Inductive Miner

The Inductive Miner is another process mining algorithm [3] used to discover process models from event logs. Like the Heuristic Miner, the Inductive Miner aims to automatically construct a process model based on observed sequences of activities in the event log. The Inductive Miner is characterized by its inductive approach, which involves building a model by generalizing from specific examples.

Here are the key characteristics of the Inductive Miner algorithm:

- Inductive Approach: The Inductive Miner employs an inductive reasoning approach, generalizing patterns from observed behavior to construct a process model.

- Directly-Follows Graph: The algorithm often uses a Directly-Follows Graph (DFG) as an intermediate representation. The DFG captures relationships between activities based on their sequential order in the event log.

- Split and Join Patterns: The Inductive Miner can identify and represent split and join patterns in a process. Split patterns occur when one activity is followed by multiple activities, and join patterns occur when multiple activities converge to a single activity.

- Noise Tolerance: The algorithm is designed to handle noisy event logs, allowing for variations and deviations in the observed sequences of activities.

- Parallelism Handling: The Inductive Miner is capable of handling parallelism in the process by recognizing concurrent execution of activities.

- Loop Handling: The algorithm can represent loops or cycles in the process, capturing repetitive patterns observed in the event log.

The Inductive Miner is known for its ability to generate more precise process models compared to some other process mining algorithms. However, the resulting models can sometimes be more complex.

Chapter 4

Process Model Creation and Use Case Description

In this chapter, we delve into the process of creating process models using the heuristic miner and the inductive miner on the generated dataset. Specifically, we first apply these mining techniques to the entire dataset, and then we partition the dataset into two subsets: one comprising weekdays (Monday to Friday) and the other consisting solely of weekends (Saturday and Sunday). This decision is driven by the expectation that the behavior of the elderly may vary between weekdays and weekends. Failing to account for this difference could result in the generation of Petri Nets that are potentially less useful, as they would be constructed based on a significantly different dataset.

4.1 Process Model Creation

1. **Mining on the Entire Dataset**: We begin by applying both the heuristic miner and the inductive miner to the complete dataset. This allows us to capture the overall process behavior without considering any temporal distinctions.

2. **Partitioning the Dataset**: Next, we divide the dataset into two distinct subsets based on the days of the week: weekdays (Monday to Friday) and weekends (Saturday and Sunday). This segregation enables us to differentiate between the behavior patterns exhibited during regular weekdays and those observed on weekends.

3. **Mining on Weekday and Weekend Subsets**: We then apply the heuristic miner and the inductive miner separately to each subset of the dataset. By doing so, we can uncover process models that are tailored to the specific behavior patterns characteristic of weekdays and weekends.

Motivation for Dataset Partitioning

The decision to partition the dataset into weekday and weekend subsets is motivated by the understanding that elderly individuals may engage in different activities and follow distinct routines depending on the day of the week. For example, on weekdays, they may adhere to structured schedules involving medical appointments or recreational activities organized by the nursing home. On the other hand, weekends may afford them more leisure time and opportunities for social interactions with family members or friends.

Disregard the differences in behavior between weekdays and weekends could lead to the generation of process models that inadequately represent the underlying processes. By explicitly accounting for these variations, we aim to create more accurate and actionable process models that better reflect the reality of elderly care in a smart nursing home environment.

The application of the heuristic miner and the inductive miner to both the entire dataset and its weekday and weekend subsets enables us to capture and analyze the nuanced behavior of elderly individuals in different contexts. This nuanced understanding is essential for developing effective process models that can improve decision-making processes. The differences between the two algorithms, in terms of results, will be explained in the next chapter. We report, now, two Petri Nets that represent the behavior of the elderly during the weekend, one obtained with the heuristic algorithm (Figure 4.1) and the other with inductive algorithm (Figure 4.2).

4.1.1 Comparison between Petri Net

When comparing the Petri Nets obtained through the heuristic algorithm and the inductive algorithm, significant differences in the representation of concurrent activities in the process emerge. These differences stem from the distinct approaches taken by the two algorithms in analyzing event data.

Petri Net obtained with the Heuristic Algorithm The heuristic approach tends to simplify the process representation by focusing on the most frequent or common activities in the event log. Consequently, the resulting Petri Net may show a lower number of concurrent activities, as it only considers activities that occur with a certain frequency or regularity.

Petri Net obtained with the Inductive Algorithm The inductive approach tends to be more inclusive, considering all activities present in the event log, including those less frequent or rare. As a result, the resulting Petri Net may exhibit a more substantial presence of concurrent activities, as it takes into account all possible activity sequences in the process, regardless of their frequency. Rare or less common activities are treated the same as more frequent activities, thereby maintaining a high degree of concurrency in the Petri Net.

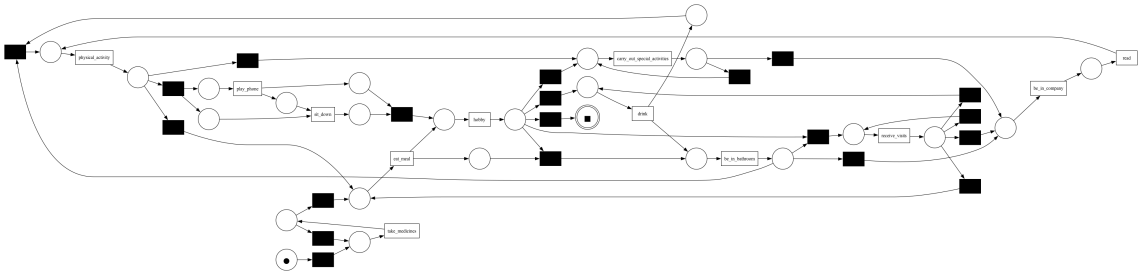

Figure 4.1: In figure there is an Heuristic Petri Net obtained from the analysis of the weekends of all the elderly.

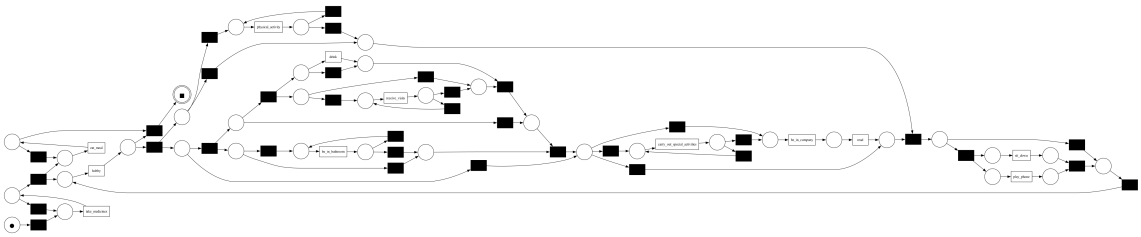

Figure 4.2: In figure there is an Inductive Petri Net obtained from the analysis of the weekends of all the elderly.

Implications The Petri Net obtained with the heuristic algorithm may be more simplified and focused on the predominant activities in the process. This could make it easier to interpret and use in contexts where a high-level representation of the process is required. On the other hand, the Petri Net obtained with the inductive algorithm may be more detailed and inclusive, reflecting the complexity and variability of the process as a whole. This could be useful for in-depth analysis and identification of less obvious patterns or anomalies. The choice between the two approaches depends on the specific analysis needs and the characteristics of the process under examination.

In the context of our project, as we will examine shortly, depending on the metric considered, one algorithm is preferable to the other.

4.2 Use Case Description

In this section, we simulate a use case scenario involving a log generated by ChatGPT containing actions performed by an elderly on a Monday. The purpose of this use case is to test the system's learning capability, specifically the ability of the Petri Net to grow as unknown actions are identified. The simulated log includes two unknown activities: "drink_coffee" and "suffer_physical_abuse". Several experiments are conducted on this log.

4.2.1 Experiment 1: Evaluation of Pre-Trained Petri Net

In the first experiment, we evaluate the Petri Net trained using only the Mondays in the synthetic dataset compared to the simulated Monday. This allows us to assess the performance of the existing Petri Net in capturing the activities observed on Mondays.

4.2.2 Experiment 2: Detection and Management of Unknown Activities

In the second experiment, the system detects the two unknown activities and filters them, separating the dangerous activities from the legitimate ones. The system then prompts the user to select which of the legitimate activities they would like to add to the set of stored activities[1]. The management of dangerous activities is delegated to a dedicated module that handles the situation at runtime. Subsequently, the system adds only the confirmed activities to the log, resulting in the growth of the Petri Net.

4.2.3 Experiment 3: Petri Net Generation and Evaluation

Following the addition of the confirmed activities to the log, a new Petri Net is generated, containing all previously known activities extended with the newly learned activities. This updated Petri Net is then evaluated on the simulated Monday log to assess its effectiveness in capturing the observed activities, including the newly learned ones.

4.2.4 Conclusion

Through these experiments, we demonstrate the system's ability to adapt and learn from new data, dynamically growing the Petri Net to accommodate previously unknown activities. This adaptive approach enhances the system's ability to accurately represent and analyze the behavior of elderly.

[1]The decision on what actions to add can be left to the user periodically. We have assumed that this is requested to the user every night, then once a day, so that the user can easily retrace the day just ended and decide what actions to add to the system.

Chapter 5

Results and Conclusions

In this chapter, we analyze the results obtained from the experiments described in the previous chapter, focusing specifically on the Petri Nets that concern all elderly individuals and the results obtained for the simulated use case.

The evaluation of the Petri Nets generated from the experiments reveals several noteworthy findings:

1. Pre-Trained Petri Net Performance: The pre-trained Petri Net, based on Mondays from the synthetic dataset, demonstrates a reasonable level of accuracy in capturing the observed activities. However, it may lack specificity in representing less common or sporadic actions.

2. Adaptive Petri Net Growth: Through the dynamic addition of newly learned activities, the Petri Net expands over time, incorporating a broader range of behaviors and enhancing its capacity to model the real-world processes accurately.

5.1 Results Analysis

The Table 5.1 presents the fitness and precision metrics obtained by applying both heuristic and inductive Petri Nets to the original dataset described in § 2.2. These Petri Nets were built on four training sets: "All Week," "Weekdays," "Weekend," and "All Mondays," obtained by considering specific portions of the synthetic dataset.

In particular, the first dataset contains logs of all elderly individuals throughout the entire week. The second dataset includes logs of all elderly individuals only for the days from Monday to Friday. The "Weekend" dataset encompasses logs of all elderly individuals for only the holiday days (Saturday and Sunday). Lastly, the final dataset is derived from logs pertaining only to Mondays present within the synthetic dataset.

From the Table 5.1, it is evident that the inductive algorithm performs best in terms of fitness, while the heuristic algorithm is better in precision.

Train Set	Inductive Log Fitness	Inductive Precision	Heuristic Log Fitness	Heuristic Precision
All Week	0.970253	0.107143	0.400185	0.333333
Weekdays	0.981761	0.100667	0.477484	0.333333
Weekend	0.273585	0.0	0.065882	0.000000
All Mondays	0.987588	0.416667	0.295470	0.500000

Table 5.1: The "Train Set" column indicates the portion of dataset used to build the Petri Net. Each line reports the results obtained by applying each Petri Net to the original dataset.

For the goal of this project, the metric with greater significance is fitness. Indeed, the model constructed will need to operate in an adaptive environment, where the set of activities is continuously expanding. Consequently, a model with higher adaptability proves to be more suitable for this purpose. Conversely, optimizing for precision implies a focus on accurately predicting known behaviors based on the limited set of activities used to train the model. While this approach can yield highly accurate predictions within the scope of the training data, it inherently restricts the model's ability to generalize to unseen activities. Since the model has not been exposed to a diverse range of activities during training, it lacks the capability to recognize and predict actions that fall outside the predefined set.

The choice between prioritizing fitness or precision depends on the specific requirements and objectives of the application. In this project, where the emphasis is on adapting to evolving environments and accommodating new activities, prioritizing fitness aligns with the overarching goal of ensuring the model's effectiveness in capturing and representing the dynamic behavior of elderly individuals in a smart nursing home setting. This approach enables the model to flexibly incorporate new activities as they are encountered, thereby enhancing its adaptability and utility over time.

In Table 5.1, we also observe that the dataset associated with the model exhibiting the poorest fitness is the "Weekend" dataset, constructed using logs from weekends. This behavior is easily justifiable when considering that during weekends, the elderly tend to engage in different activities compared to their weekday routine. Moreover, these activities may occupy a significant portion of the day, such as visits from relatives or extraordinary activities organized by the facility (such as outings).

During weekends, individuals often have more leisure time and may participate in social or recreational activities that differ from their weekday obligations. These deviations from the regular routine can introduce variability and complexity into the behavior patterns observed during weekends, making it more challenging for the model to accurately capture and predict these activities.

In the first row of Table 5.2, we present the results obtained from the model constructed on the "All Mondays" dataset applied to the log associated with the

Test Set	Inductive Log Fitness	Inductive Precision	Heuristic Log Fitness	Heuristic Precision
Simulated Monday	0.960520	0.000000	0.712211	0.25
Filtered Monday	0.987671	0.000000	0.829971	0.000000

Table 5.2: This table refers to the Petri Net trained only on "All Mondays Train Set". In the first line it is applied to the simulation of a Monday. In the second line it is applied to a filtered Monday, i.e. without violent actions.

scenario described in § 4.2. In the second row, we report the results obtained from the model built on the "All Mondays" dataset applied to the filtered log obtained from Experiment 3 as described in § 4.2.3.

From the results, it is evident that the model exhibits excellent initial fitness. It further improves as the user accepts to add new activities to their routine. The model's strong initial fitness indicates its effectiveness in capturing and representing the observed behavior patterns. As the model encounters new activities and adapts to the user's preferences, its fitness continues to improve, reflecting its ability to dynamically learn and evolve over time. Therefore, the objective of this project appears to be achieved.

5.2 Conclusions

In conclusion, we have demonstrated the feasibility of incrementally growing a Petri Net as knowledge is acquired from real-world data, bridging process mining techniques. By integrating learning capabilities into the process mining framework, we enable the system to adapt and evolve in response to new information, thereby improving its ability to accurately represent and analyze complex processes. This adaptive approach holds significant promise for enhancing our understanding of elderly behavior in smart nursing home environments, ultimately leading to improved care delivery and safety measures.

5.3 Future Enhancements

Future enhancements can be seen both from the design and implementation perspective.

5.3.1 Design Enhancements

In terms of design, future developments could focus on enriching the prompts provided to ChatGPT to generate a more diverse and realistic dataset. This could

involve refining the prompts to capture a broader range of activities and scenarios that reflect the complexities of real-world environments. For example, including prompts that simulate interactions between elderly individuals and caregivers, or prompts that incorporate environmental factors such as weather conditions or facility layout.

Additionally, efforts could be made to enhance the diversity and complexity of the generated dataset by introducing more nuanced variations in activity sequences and timings.

5.3.2 Implementation Enhancements

From an implementation perspective, future developments could involve leveraging real-world datasets of recognized actions, potentially captured through personal devices such as smartwatches or bracelets. Integrating such datasets with the simulation could provide a richer and more accurate representation of elderly behavior in a smart nursing home environment.

Special emphasis could be placed on sensitive activities such as medication intake or instances of violence. By incorporating data from personal devices equipped with sensors capable of detecting specific actions or behaviors, such as taking medication or physical aggression, the simulation could better reflect the challenges associated with managing elderly care in real-world settings.

Future developments in both design and implementation could significantly enhance the realism and effectiveness of simulations of elderly behavior in smart nursing home environments. By incorporating more diverse and comprehensive datasets and leveraging emerging technologies we can better understand and address the complex dynamics of elderly care.

Bibliography

[1] Nutchar Senewong Na Ayutaya, Prajin Palungsuntikul, and Wichian Premchaiswadi. "Heuristic mining: Adaptive process simplification in education". In: *2012 tenth international conference on ict and knowledge engineering*. IEEE. 2012, pp. 221–227.

[2] Alessandro Berti, Sebastiaan van Zelst, and Daniel Schuster. "PM4Py: a process mining library for Python". In: *Software Impacts* 17 (2023), p. 100556.

[3] Ifrina Nuritha and E.R. Mahendrawathi. "Structural Similarity Measurement of Business Process Model to Compare Heuristic and Inductive Miner Algorithms Performance in Dealing with Noise". In: *Procedia Computer Science* 124 (2017). 4th Information Systems International Conference 2017, ISICO 2017, 6-8 November 2017, Bali, Indonesia, pp. 255–263. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.12.154. URL: https://www.sciencedirect.com/science/article/pii/S1877050917329228.