Immagine che contiene emblema, cresta, Marchio, simbolo

Descrizione generata automaticamente

Formal Methods 2024/2025 ​Federica Picca -​Alberta Motca Schnabel

**exercise 5: PROCESS MINING  
- Tool Prom and python library pm4py + LLM Integration**

**WHAT IS PROCESS MINING ?**

Process mining combines techniques from data science and business process management to analyze, monitor, and improve processes based on data recorded by information systems. The goal is to discover how business processes are actually executed, identify inefficiencies, bottlenecks, deviations, and provide insights useful for optimizing operations.

In our work, we have concentrated on the **Process Discovery** field, particularly utilizing the **Petri net model**.

**WHAT IS PROCESS DISCOVERY ?**

Process discovery is the technique of constructing a process model from an event log. Event logs are structured records of events that have occurred in a process.

**WHAT IS A PETRI NET ?**

**Petri nets** are a mathematical representation of processes that provide a robust framework for modeling and analyzing the flow of activities. By leveraging Petri nets, we can create detailed and precise process models from event logs, capturing the complexities and dependencies within processes. This approach allows for a more granular and accurate depiction of workflows, enabling better analysis and optimization.

**Soundness**

Soundness refers to whether the Petri Net model correctly represents the process it is supposed to model. A sound Petri Net is one where:

* Every valid trace in the process can be completed without deadlocks or unreachable states.
* It doesn't allow for infinite loops or incomplete process executions.
* It guarantees that any process described by the log can start, proceed, and end correctly, following the rules of the Petri Net.

In essence, a sound Petri Net ensures that the modeled process is logically consistent and can handle real-life process execution without issues.

**Workflow Net**

A workflow net is a special type of Petri Net designed to represent workflows in a structured manner. It is defined by:

* A single start place (representing the beginning of the process).
* A single end place (representing the completion of the process).
* The net must be sound, meaning there are no deadlocks, and every process instance can reach the end place from the start place.

A workflow net focuses on modeling workflows where the goal is to describe the execution flow from start to end, ensuring that every task leads to a conclusion.

**1. Fitness**

Fitness measures how well the discovered Petri Net can reproduce the behavior observed in the event log. A model with high fitness allows all or most of the traces in the log to be replayed without deviations.

* **High fitness** indicates that the model captures the behavior recorded in the log effectively.
* **Low fitness** suggests that the model fails to replicate some behaviors from the log, meaning certain traces cannot be properly executed or represented by the Petri Net.

**2. Simplicity**

Simplicity assesses the complexity of the generated Petri Net. It aims to ensure that the model is not overly complicated or unnecessarily detailed.

* A **simple model** is preferred as it is easier to interpret and analyze.
* Overly **complex models** with many redundant elements or intricate structures are penalized, even if they fit the log well, to avoid overfitting.

**3. Precision**

Precision evaluates whether the Petri Net generates behavior that is consistent with the observed log. It ensures that the model does not allow for behaviors that are not present in the log.

* **High precision** means the model restricts its behavior to what is recorded in the log, avoiding overgeneralization.
* **Low precision** indicates that the model permits traces or transitions that were not observed, which can lead to misleading conclusions.

**4. Generalization**

Generalization measures how well the Petri Net can represent new, unseen behavior that is consistent with the observed process. It balances the trade-off between precision and flexibility.

* **Good generalization** ensures that the model captures the essential characteristics of the process and can accommodate variations or slight changes in behavior.
* A model that is too rigid may **lack generalization**, while one that is too flexible might lose precision.

These metrics often need to be balanced as they can conflict with one another. For example:

* Improving fitness may require adding more elements, which could reduce simplicity.
* Enhancing precision might decrease generalization, as a more restrictive model may fail to capture new valid behaviors.

**DATASET**

The **kasterenC\_dataset** is a dataset commonly used in process mining or activity recognition studies. It represents sequences of activities or events recorded in a smart home, such as cooking, cleaning, watching TV, or sleeping.

The dataset contains event logs capturing the occurrence of activities over time. Each event is associated with specific attributes, such as timestamps, activity types, or sensor IDs.

It is structured in a tabular format CSV, where each row represents an event or activity instance, including details like:

* + **Timestamp**: the time when the activity occurred.
  + **Activity name**: the specific action or event being recorded.
  + **Sensor**: the identifier or type of sensor that recorded the activity, indicating the source of the data
  + **Start/end**: The indicators marking the beginning and completion of an activity. "Start" refers to the moment when an activity begins, and "End" refers to the moment when the activity finishes. These may help track the duration of an activity.

The data are collected from real-world smart homes equipped with a variety of sensors.

The types of sensors typically used in this dataset include:

* **ToiletFlushDownstairs**: this sensor detects when the toilet is flushed in the downstairs bathroom, capturing an event when the toilet's flushing mechanism is activated.
* **Fridge**: this sensor monitors activity related to the fridge, such as the opening or closing of the door, or possibly the temperature inside, indicating usage or access to the fridge.
* **PlatesCupboard**: this sensor tracks activity related to the cupboard where plates are stored. It may detect events like the opening of the cupboard or the removal or placement of plates.
* **Couch**: this sensor detects activity around the couch, such as someone sitting on it, getting up, or even detecting pressure from seating.
* **BedRight**: this sensor monitors the right side of the bed. It could detect when someone gets into or out of bed, or other activities like adjusting the bed's position.
* **ToiletFlushUpstairs**: tracks toilet flushing events in the upstairs bathroom, signaling when the toilet has been flushed.
* **BedLeft**: this sensor tracks activity on the left side of the bed, monitoring events such as someone lying down or getting up.
* **Dresser**: this sensor monitors activity around the dresser, detecting actions such as drawers opening and closing, or changes in position of the items stored in the dresser.
* **Cutlery**: this sensor detects activities related to cutlery, such as the removal or placement of cutlery in a drawer or utensil holder, or possibly the use of cutlery for dining.
* **Frontdoor**: this sensor monitors events involving the front door, such as door opening or closing, indicating when someone enters or exits the home.
* **Saucepan**: this sensor detects activity around the saucepan, such as being used on the stove or its removal from a cupboard, potentially related to cooking actions.
* **SinkTop**: this sensor tracks events on the surface of the sink, like washing dishes, placing items on the countertop, or other activities related to the sink area.

**TOOL: ProM**

ProM is a process mining tool used for analyzing process logs, event sequences, and discovering process models.

When the dataset is imported into ProM, it's likely transformed, through the plugin *Convert CSV to XES***,** into an event log format that ProM can work with (in this case **XES -** eXtensible Event Stream format). These logs represent events and their sequence over time.

Additionally, through ProM it is possible to obtain a visualization of an event log, showing a statistical analysis of the recorded data.

**Dataset Overview**

* **Total number of process instances**: 21  
  This means the dataset includes 21 unique cases or processes. Each case represents a sequence of events executed in a specific order.
* **Total number of events**: 45,400  
  This indicates the total number of individual events recorded across all instances.

**Event Classes**

**Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamente**

The dataset defines 18 event classes. These are the distinct types of activities captured in the process. The most frequent classes include:

1. **GoToBed**:
   * Occurrences (absolute): 36,578
   * Percentage: 80.568%
2. **None**:
   * Occurrences (absolute): 3,366
   * Percentage: 7.414%
3. **TakeShower**:
   * Occurrences (absolute): 1,920
   * Percentage: 4.229%

Other less frequent event classes include **Relax**, **PrepareDinner**, **BrushTeeth**, and more.

**Start and End Events**

**Immagine che contiene testo, schermata, schermo, numero

Descrizione generata automaticamente**

* **Start Events**:  
  There are 11 distinct classes of start events. The most frequent are:
  + **None (23.81%)**
  + **BrushTeeth (14.286%)**
  + **PrepareDinner (14.286%)**
* **End Events**:  
  There are 7 distinct classes of end events. The most frequent are:
  + **None (52.381%)**
  + **Relax (14.286%)**
  + **PrepareDinner (9.524%)**

**Lifecycle Transitions**

**Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente**

Immagine che contiene testo, schermata

Descrizione generata automaticamente

Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamente

The dataset includes event classes defined by a combination of **event name** and **lifecycle transition** (e.g., GoToBed+start, GoToBed+complete).

* **Total classes (event name + lifecycle transition)**: 36
* Examples:
  + GoToBed+start: 18,289 occurrences (40.284%)
  + GoToBed+complete: 18,289 occurrences (40.284%)

**Most Frequent Activity**: The dataset is dominated by the event class **GoToBed**, making up over 80% of all occurrences.

**Event Lifecycle**: Many events have both start and complete lifecycle transitions, suggesting detailed tracking of each activity's lifecycle.

**ANOTHER VISUALIZATION OF STATISTICS**

Immagine che contiene schermata, Software multimediale, Software per la grafica, software

Descrizione generata automaticamente

**Events per Case:** the graph shows a much wider variation (minimum 12 events, average 2162, maximum 37434), indicating that the cases in the dataset are many and complex, with a higher number of events associated with each case.

**Event Classes per Case:** Here too, there is a significant diversification (average 16 classes, maximum 30), suggesting a greater variety in the activities and types of events recorded.

In the dataset described, the presence of activities labeled as **"None"** is an interesting detail to analyze.

The label **"None"** could represent:

* **Missing events**: It may indicate that certain activities were not recorded correctly in the log, leaving the activity name field empty.
* **Unclassified activities**: It could act as a placeholder for activities that do not fit predefined categories or were not recognized during data recording.
* **Pauses or idle states**: In some contexts, "None" might represent periods of inactivity or transitions between meaningful activities.
* **Acquisition or preprocessing errors**: If the log was preprocessed, "None" might have been added for incomplete activities or corrupted data.

With **3,366 occurrences (7.414%)**, "None" represents a significant portion of the log.

**Initial Application of Alpha Miner and Inductive Miner**

An initial analysis was conducted on the complete dataset. As a first step, starting from the dataset in XES format, the action *Mine for a Petri Net using Alpha-algorithm* was applied. Specifically, by selecting the Alpha Miner and Inductive Miner algorithms, the associated Petri Nets were generated.

**ALPHA MINER**

Immagine che contiene schizzo, diagramma, linea, disegno

Descrizione generata automaticamente

**Filtering the Dataset**

The dataset is then transformed into a format suitable for process mining, where each **trace** represents a series of activities or events performed by an individual in the smart home.

In our case, the filtering is performed according to the following criteria:

* removing logs containing only "end" or only "start" (thus keeping only completed activities)
* eliminating activities labeled as "None”.

The plugin used in this case is *Filter Log using Simple Heuristics.*

This preprocessing highlights like removes unnecessary or incomplete information, making the dataset ready for advanced process mining tasks.

The filtered dataset contains the following characteristics:

Immagine che contiene schermata, testo, software, Parallelo

Descrizione generata automaticamente

Immagine che contiene schermata, testo, software, Parallelo

Descrizione generata automaticamente

**Log Summary:**

* **Total number of process instances:** 9
* **Total number of events:** 774

**Event Classes:**

* The dataset includes **4 event classes**:
  1. **Relax:** 244 occurrences (31.525%)
  2. **PrepareDinner:** 206 occurrences (26.615%)
  3. **BrushTeeth:** 188 occurrences (24.289%)
  4. **GetDressed:** 136 occurrences (17.571%)

**Start Events:**

* **Total classes of start events:** 4
  + **BrushTeeth:** 3 occurrences (33.333%)
  + **PrepareDinner:** 3 occurrences (33.333%)
  + **Relax:** 2 occurrences (22.222%)
  + **GetDressed:** 1 occurrence (11.111%)

**End Events:**

* **Total classes of end events:** 4
  + **Relax:** 3 occurrences (33.333%)
  + **PrepareDinner:** 3 occurrences (33.333%)
  + **GetDressed:** 2 occurrences (22.222%)
  + **BrushTeeth:** 1 occurrence (11.111%)

Once the dataset is loaded into ProM, various preprocessing actions can be performed. This can include:

* **Cleaning the data**: Removing irrelevant or incomplete data
* **Event filtering**: Applying filters to focus on specific types of events
* **Handling noise**: Removing noise or irrelevant information, such as inactive states.

**ANOTHER VISUALIZATION OF STATISTICS**

Immagine che contiene schermata, Software multimediale, Software per la grafica, software

Descrizione generata automaticamente

**Key Data:**

* **Processes:** 1  
  The dataset contains a single process.
* **Cases:** 9  
  There are 9 process instances in the log.
* **Events:** 774  
  A total of 774 events are recorded across all cases.
* **Event Classes:** 4  
  Four distinct event classes are identified in the dataset: "Relax," "PrepareDinner," "BrushTeeth," and "GetDressed."
* **Event Types:** 1  
  Only one type of event transition ("complete") is present.
* **Originators:** 1  
  All events are attributed to a single originator or source.

**Histograms:**

**Top Histogram (Events per Case):**

* This bar chart shows the distribution of the number of events across the 9 cases:
  + **Minimum:** 6 events in the smallest case.
  + **Mean:** On average, each case has 86 events.
  + **Maximum:** The largest case contains 248 events.
  + The chart demonstrates a wide variability in the number of events per case, with the majority of cases clustered near the higher end of the range.

**Bottom Histogram (Event Classes per Case):**

* This bar chart illustrates the number of distinct event classes observed in each case:
  + **Minimum:** 1 event class in the simplest case.
  + **Mean:** Each case, on average, has 2 distinct event classes.
  + **Maximum:** The most complex cases contain all 4 event classes.
  + The chart shows a significant number of cases with more event classes, as indicated by the larger bars toward the higher end of the distribution.

**Application of Alpha Miner and Inductive Miner to Filtered Dataset**

The frist step is to load XES file in ProM and select one of the algorithms to generate a Petri Net.

From the available menu, you can choose a process mining algorithm such as:

* **Alpha Miner**: This is one of the simplest and most well-known algorithms for process discovery. It is particularly useful for discovering a Petri Net when the event log is clear and relatively noise-free. Alpha Miner assumes the log contains sufficient information to directly model the process without significant inconsistencies or missing data.
* **Heuristic Miner**: This algorithm is more robust and effective when working with noisy, incomplete, or imperfect datasets. Heuristic Miner doesn't require as much structure in the event log and can handle variations in event ordering and missing activities. It is often the preferred choice when the log data is not perfectly structured or when noise is present, as it can still produce a meaningful Petri Net representation.
* **Inductive Miner**: This algorithm is ideal for datasets that are well-structured but may be more complex in nature. Inductive Miner is well-suited for logs with more intricate process models and can effectively discover the underlying Petri Net while ensuring that the discovered model is sound and conforms to the event log data. It excels with structured logs that may contain many activities and complex behavior.

After selecting the appropriate algorithm based on the nature of the dataset, you apply it to the event log. ProM will then process the log according to the chosen algorithm's rules and methods.

Once the algorithm has been executed, ProM will generate a Petri Net model that reflects the underlying process flow, including places, transitions, and arcs, based on the patterns and relationships detected in the filtered dataset.

The Petri Net created by applying the Alpha Miner algorithm was generated using the *Alpha Miner* plugin, while the one created by applying the Inductive Miner algorithm was generated using the *Mine Petri Net with Inductive Miner* plugin and the one created by applying the Heuristic Miner algorithm was generated using frist *Mine for a Heuristic Net Using Heuristic Miner* plugin, and later *Convert Heuristic Net to Petri Net* plugin.

**ALPHA MINER**

**Immagine che contiene diagramma, linea, schizzo, testo

Descrizione generata automaticamente**

**Chat GPT Analysis:**

**Immagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamente**

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

**INDUCTIVE MINER**

**Immagine che contiene schizzo, diagramma, disegno, Line art

Descrizione generata automaticamente**

**Chat GPT Analysis:**

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

**HEURISTIC MINER**

**Immagine che contiene diagramma, linea, Piano, schizzo

Descrizione generata automaticamente**

**Chat GPT Analysis:**

**Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente**

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

**Evaluation of Metrics**

First, it was verified that the Petri Nets generated by the algorithms are both sound and meet the criteria of a workflow net. This verification was performed using the *Analyze with Woflan* plugin.

Subsequently, metrics were employed to evaluate the quality of the Petri Nets generated through process mining algorithms.

To calculate these metrics, the filtered dataset file in XES format and the Petri Net generated by applying the Inductive Miner algorithm were selected. The plugin *Replay a Log on Petri Net for Conformance Analysis* was then utilized to calculate the fitness of the model

Fitness measures how well the Petri Net can reproduce the observed behavior in the log.

* A fitness score close to 1.0 means the model perfectly captures the log’s behavior.
* Lower fitness indicates misalignment, meaning the Petri Net fails to represent certain parts of the process.

Next, use the *Measure Precision/Generalization* plugin on the previous result. This step assesses two additional aspects of the Petri Net's quality: **precision** and **generalization**.

Precision ensures that the Petri Net doesn’t allow behaviors that weren’t observed in the log.

A high precision score means the model is restrictive and closely aligned with the log.

Generalization measures the model's ability to handle new, valid behaviors not explicitly seen in the log.

There is no direct method to obtain the simplicity metric in ProM. Instead, the plugin *Analyze Structural Property of Petri Net* was used. This plugin provides structural properties of the Petri Net, such as:

* The number of places, transitions, and arcs in the model
* Structural relationships and characteristics that describe the complexity of the Petri Net.

Using the information provided by this plugin, you can calculate the **simplicity** metric manually by applying its specific formula. The simplicity formula typically considers factors like the number of elements (places, transitions, and arcs) in the Petri Net and evaluates whether the model is unnecessarily complex.

Where:

* P = the number of places in the Petri Net.
* T = the number of transitions in the Petri Net.
* A = the number of arcs in the Petri Net.

This formula calculates the simplicity by comparing the number of places and transitions (which represent the model’s components) to the number of arcs (which represent the connections between these components).

* A lower value of S suggests a simpler Petri Net, as there are fewer arcs compared to places and transitions.
* A higher value of S indicates a more complex Petri Net with relatively more arcs.

**Evaluation of Metrics of alpha miner petri net:**

* **Is the Petri Net a Workflow Net?**
* **Is the Petri Net sound?**

Immagine che contiene testo, Carattere, schermata, design

Descrizione generata automaticamente

* **Semplicity**

**Evaluation of Metrics of inductive miner petri net:**

* **Is the Petri Net a Workflow Net?**
* **Is the Petri Net sound?**

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

* **Fitness**

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

* **Precision**
* **Generalization**

**Immagine che contiene Carattere, testo, bianco, design

Descrizione generata automaticamente**

* **Semplicity**

**Evaluation of Metrics of heuristic miner petri net:**

* **Is the Petri Net a Workflow Net?**
* **Is the Petri Net sound?**

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

* **Semplicity**

In addition to using the ProM tool, the same operations were performed using the Python library *pm4py*. This allowed for a direct comparison of the results obtained with the two different approaches.

**PM4PY**

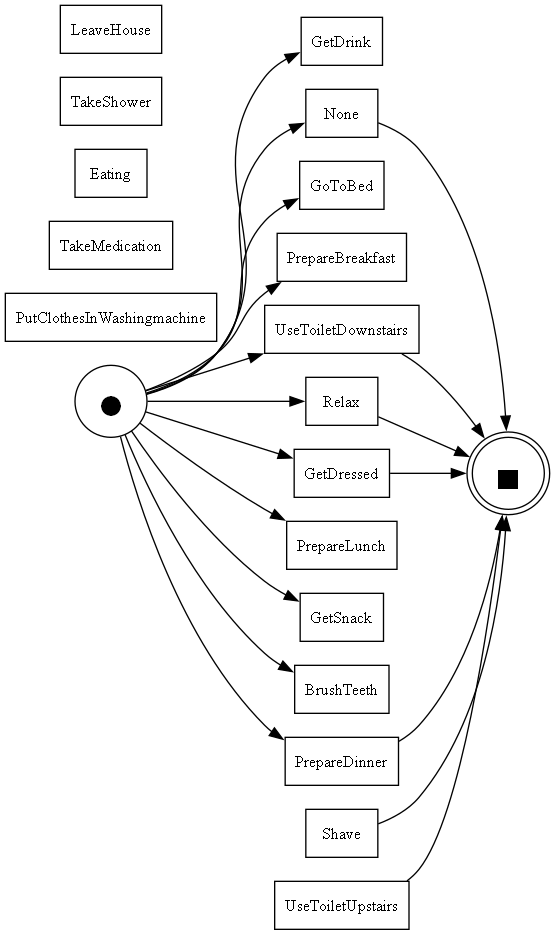
Among the myriad of tools available, the PM4Py library for Python stands out as a powerful, open-source solution. PM4Py implements a variety of process mining techniques, including process discovery, conformance checking, and model enhancement, offering users a comprehensive suite of functionalities to analyze and improve their processes.

PM4Py facilitates process discovery by transforming raw event data into visual process models. These models provide a clear and intuitive representation of how processes are executed in reality, offering valuable insights into the sequence and flow of activities.

**Process Discovery Methods Applied**

**Initial Application of Alpha Miner**

The process discovery analysis began using the Alpha Miner method. Specifically, the function *pm4py.discover\_petri\_net\_alpha* from the PM4Py library was applied to the initial dataset.

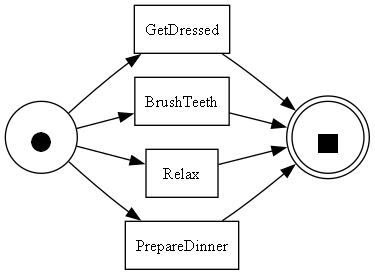


**Filtering the Dataset**

Subsequently, the Alpha Miner was applied to generate a Petri net from a filtered version of the same dataset. The original dataset was filtered using the ProM tool to exclude the "none" activity recorded and actions that had either a start but no end, or vice versa. This refined dataset allowed for a more accurate and relevant process model. It was decided to have ChatGPT comment on the obtained Petri nets.

**Application of Alpha Miner to Filtered Dataset**

After filtering, the Alpha Miner method was applied again to the filtered dataset using the same *pm4py.discover\_petri\_net\_alpha* function. This step aimed to ensure that the process models derived from the filtered data were free from irrelevant or incomplete activities.



**Chat GPT Analysis:**

Immagine che contiene testo, schermata, Carattere

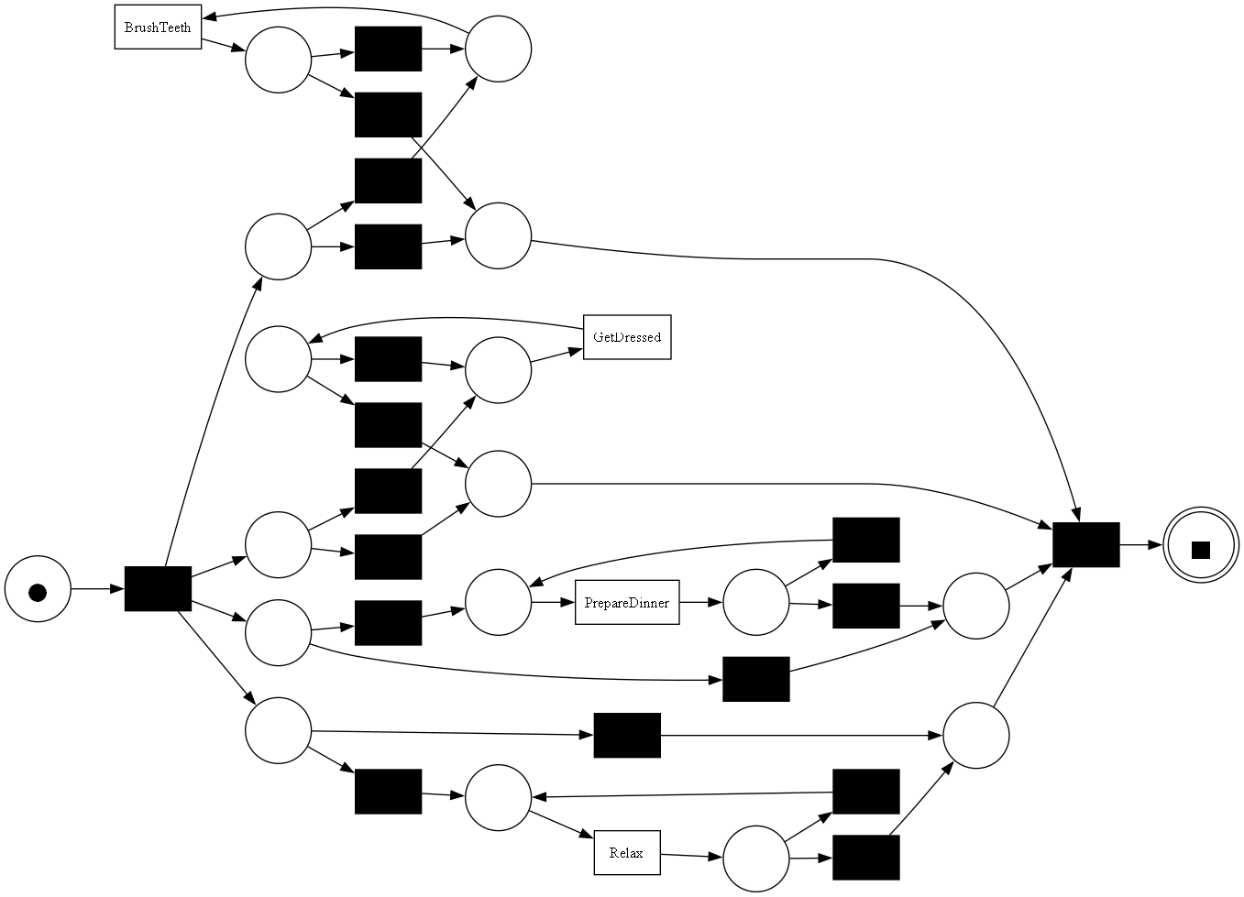
Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

**Application of Inductive Miner**

Following the application of the Alpha Miner, the **Inductive Miner** method was employed using the function *pm4py.discover\_petri\_net\_inductive(log)* on the same filtered dataset. This method provided an additional perspective on the process model, leveraging the strengths of the inductive approach.



**Chat GPT Analysis:**

Immagine che contiene testo, schermata, Carattere

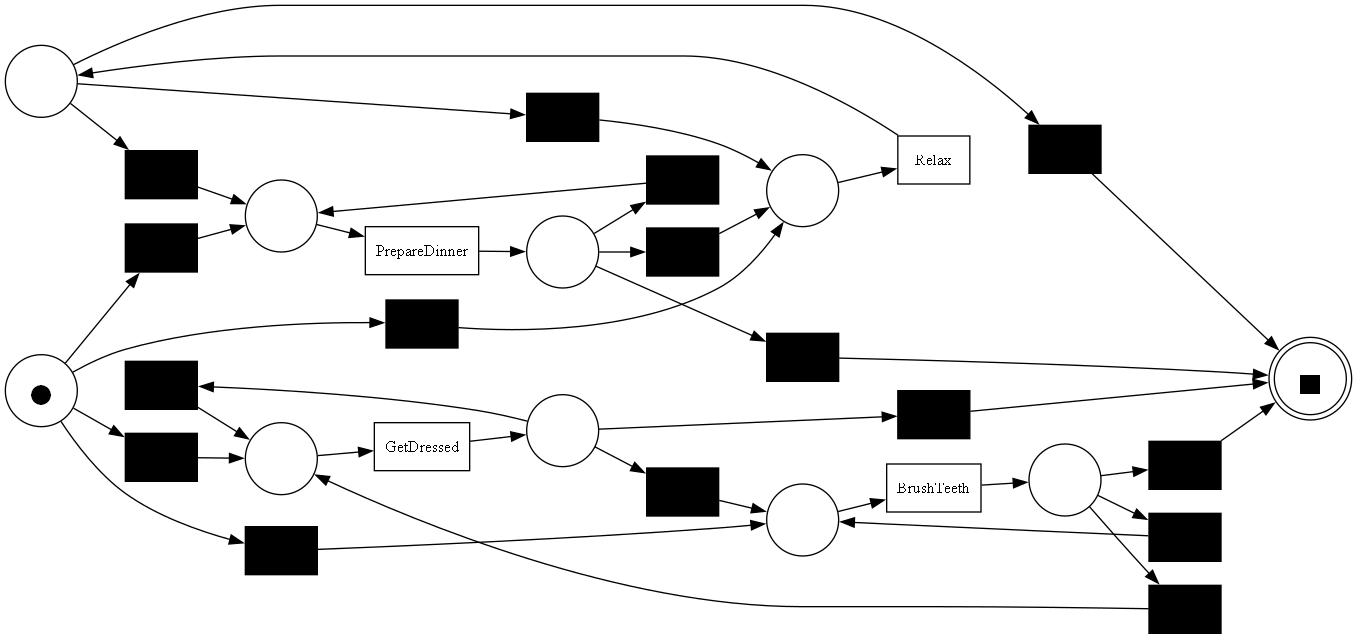
Descrizione generata automaticamente

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

**Application of Heuristic Miner**

Finally, the **Heuristic Miner** method was applied using the function *pm4py.discovery.discover\_petri\_net\_heuristics(log)* on the filtered dataset. This approach further enriched the analysis by incorporating heuristic techniques to uncover the underlying process structure.



**Chat GPT Analysis:**

**Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente**

**Evaluation of Metrics**

For each of the three Petri nets obtained from the filtered dataset—using Alpha Miner, Inductive Miner, and Heuristic Miner—soundness and workflow properties were first evaluated. Subsequently, metrics of fitness, precision, generalization, and simplicity were computed to provide a comprehensive evaluation of the process models, ensuring they are accurate, reliable, and easy to understand.

**Evaluation of Metrics of alpha miner petri net:**

* **Is the Petri Net a Workflow Net?**

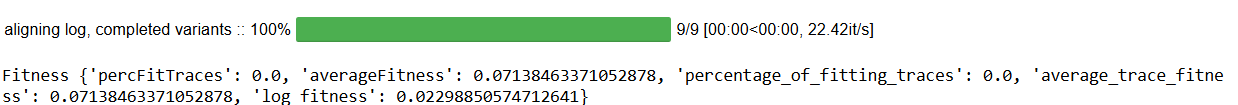


* **Is the Petri Net sound?**

Immagine che contiene testo, Carattere, schermata, documento

Descrizione generata automaticamente

* **Fitness**



* **Precision**



* **Generalization**



* **Semplicity**



**Evaluation of Metrics of inductive miner petri net:**

* **Is the Petri Net a Workflow Net?**



* **Is the Petri Net sound?**

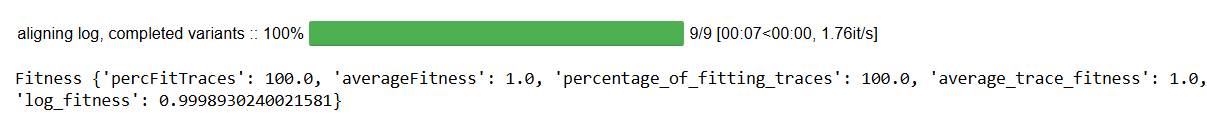
Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

* **Fitness**



* **Precision**



* **Generalization**



* **Semplicity**



**Evaluation of Metrics of heuristic miner petri net:**

* **Is the Petri Net a Workflow Net?**



* **Is the Petri Net sound?**

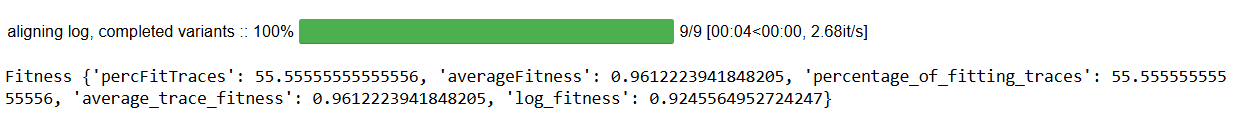
Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

* **Fitness**



* **Precision**



* **Generalization**



* **Semplicity**



**Comparison between the two methods**

**Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente**

**LLM INTEGRATION**

**FLAN T5 LARGE**

FLAN-T5 Large is a large language model based on Google's T5 (Text-to-Text Transfer Transformer) architecture, specifically fine-tuned on the FLAN (Finetuned Language Net) dataset. This tuning process enhances its ability to follow instructions and generalize across a wide variety of tasks. With approximately 780 million parameters, FLAN-T5 Large strikes a balance between performance and computational efficiency, making it suitable for many natural language processing tasks like summarization, translation, and question answering.

**Immagine che contiene testo, documento, menu, schermata

Descrizione generata automaticamente**

**PROMPT 1. And ANSWER**

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

  
  
**PROMPT 2. And ANSWER**

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente



**PROMPT 3. And ANSWER**

Immagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

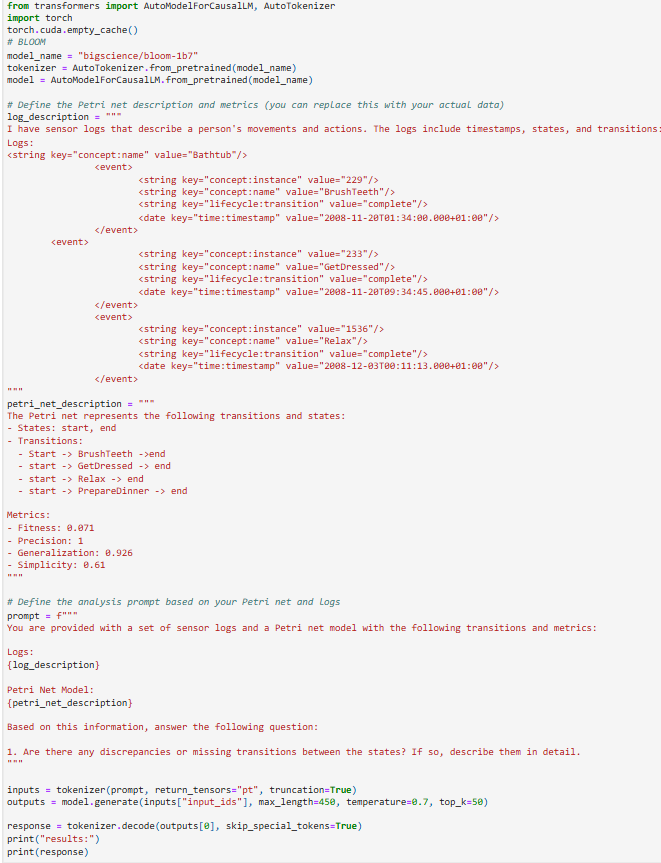
**PROMPT 4. And ANSWER**

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

**BLOOM**

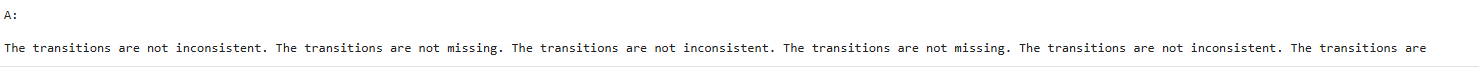
BLOOM (BigScience Large Open-science Open-access Multilingual Language Model) is a multilingual large language model developed by the BigScience project, designed to support 46 languages and 13 programming languages. It is an open-access model with billions of parameters, ranging up to 176 billion in its largest version. BLOOM is notable for its focus on inclusivity, transparency, and open collaboration, making it a widely used model in research and applications across diverse linguistic and cultural contexts.



**PROMPT 1. And ANSWER**

Immagine che contiene testo, schermata, Carattere

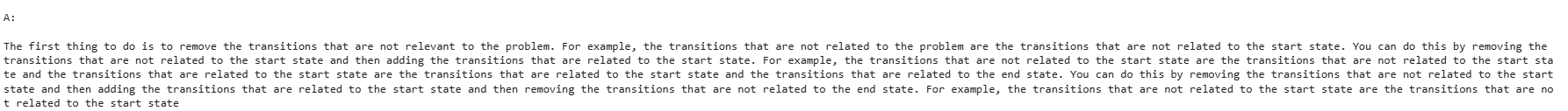
Descrizione generata automaticamente



**PROMPT 2. And ANSWER**

Immagine che contiene testo, schermata, Carattere, algebra

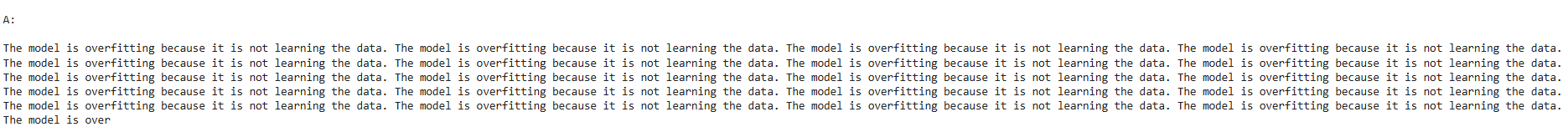
Descrizione generata automaticamente



**PROMPT 3. And ANSWER**

Immagine che contiene testo, schermata, Carattere, algebra

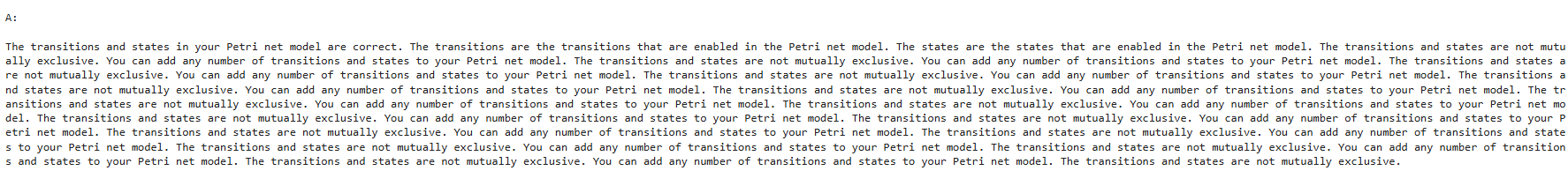
Descrizione generata automaticamente



**PROMPT 4. And ANSWER**

Immagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamente



**CHAT GPT-4**

ChatGPT was provided with the same prompts used for FLAN-5 and BLOOM. The outputs generated by ChatGPT are as follows:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente