****

**PROCESS MINING**

STUDY CASE FOR FORMAL METHODS

GIOVANNI MAGRONE

777341

Index

PROCESS DISCOVERY\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3.

PROCESS MAP\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5.

CONFORMANCE CHECKING\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7.

CONCLUSIONS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9.

PROCESS DISCOVERY

With the phase of process discovery, we want to apply algorithms to get a process model that describes the order of the event that are executed during the execution of a process

There are several mining algorithms but for our purposes we decide to use two algorithms for mining process, **HEURISTIC** and **INDUCTIVE.**

Heuristic:

- Takes frequency into account

- Detect short loops

- Does not guarantee a sound model

- Whit this algorithm is possible to get Heuristic Net and a Petri Net.

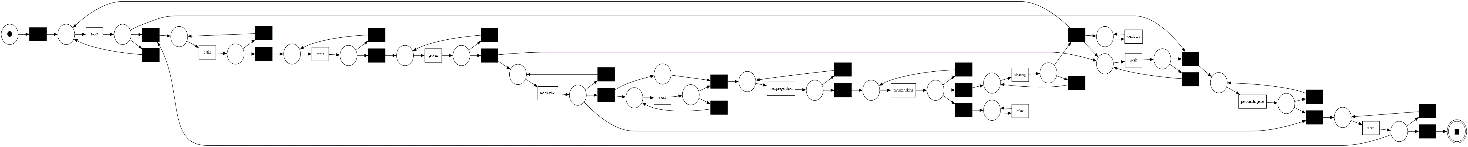


Figure 1 This is an example of PN generated with the Heuristic Miner.

Inductive:

-This model can handle invisible tasks

-Model generated is sound

-Is the most used process mining algorithm

-Two process models can be derived: Petri Net and Process Tree.

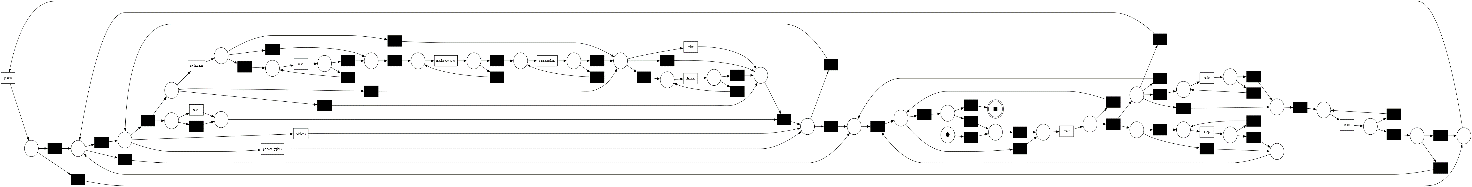


Figure 2 This is an example of PN model generated with the Inductive Miner.

After creating the Petri Net we developed two new algorithms to see how the algorithms of mining works if we want to insert new places or if an updated version of the log works with the PN generated.

The first thing we can see is that the PN works well, if we insert a new place name, that place will be correctly added to the Petri Net.

Immagine che contiene testo

Descrizione generata automaticamente

Figure 3 Inserting a new place for the PN

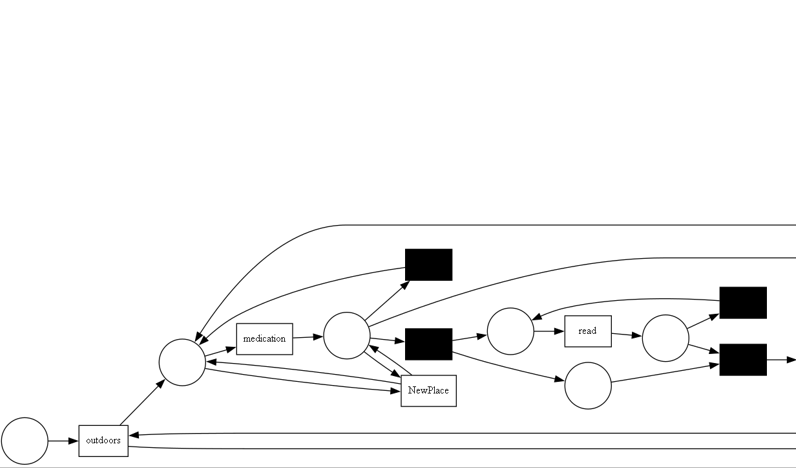
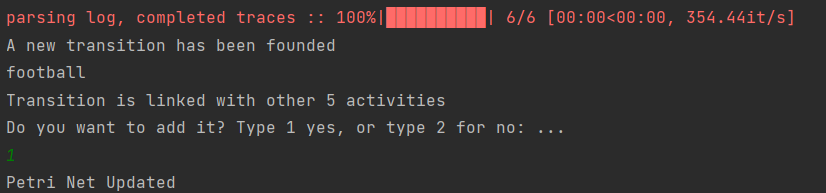


Figure 4 Showing the new place inside the PN

The second thing that we can check is that given two logs, an old one and a new one, the algorithm finds new places or transition that should be added to the Petri Net, it gives us back even the occurrences of the new Event on the log and we let the user if he wants to add it or he want to ignore it. If the log gives back a similar PN it will say that there are no differences between the two Petri Nets.



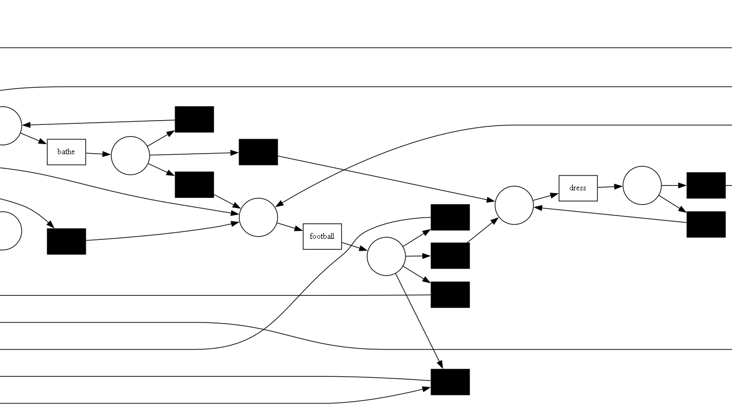


Figure 5 Example of how new Transition has been founded on the add to the PN if requested

PROCESS MAP

Not all the process mining model provide an extended support for discovering process models. Often, as a main visualization of processes, process maps are used.

A process map contains activities and connections (by means of arcs) between them. A connection between two activities usually means that there some form of precedence relation. In its simplest form, it means that the ‘source’ activity directly precedes the ‘target’ activity.

In our case we implemented the DFG Direct Follows graph from which we obtained a Process Model, this technique is the easier to do but do not give us a very well explained Process Map.

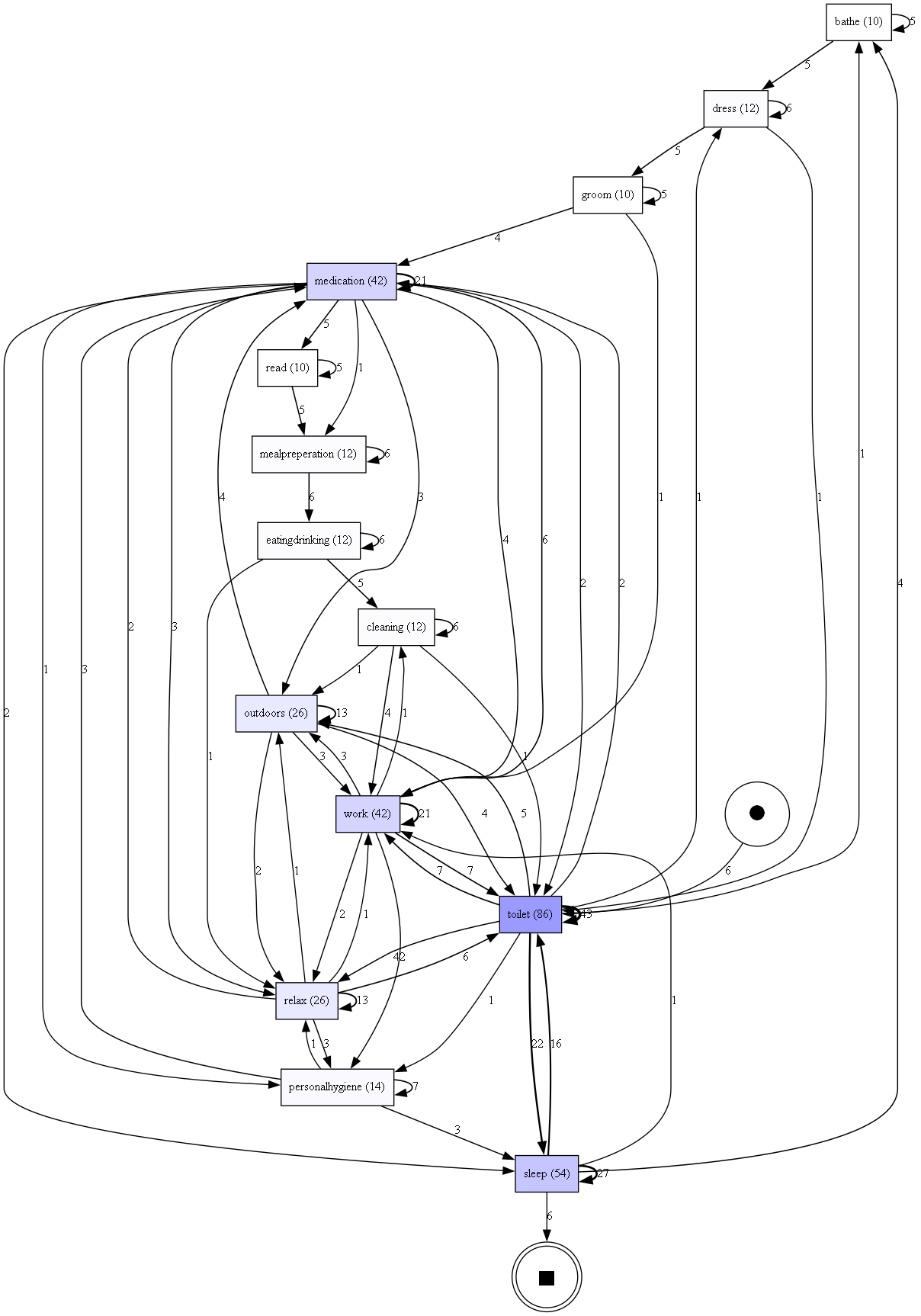


Figure 6 Example of DFG (Direct Follows Graph)

Instead, a second Process Map we obtained from the Heuristic Miner, in this case we got a very well explained and more readable Process Map for our purposes, it shows us the possible path between all activity given by execution and gives back the occurrence of the path.

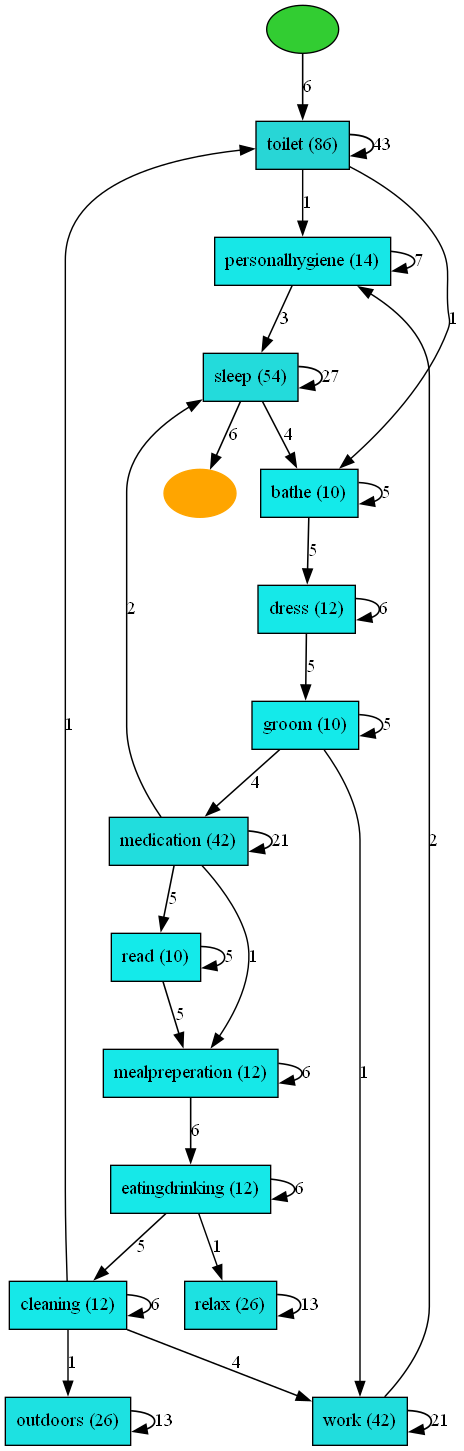


Figure 7 Example of Process Map generated from Heuristic Net

CONFORMANCE CHECKING

Now that we obtained a Process model, we want to see if our model fit with the log form which we generated it.

Conformance checking is a process to compare a process model with an event log of the same process to see if they fit each other.

In our case we are going to use the Token-Based Replay technique. TB replay matches a trace and a petri net model, starting from the initial marking seeing which transitions are executed and the tokens history of that trace.

We implemented it on the Inductive\_miner and on the Heuristic\_miner and printed the results on the console.

After the execution, we noticed that Inductive Mining gains a more correct model that fit better with the log than the Heuristic one.

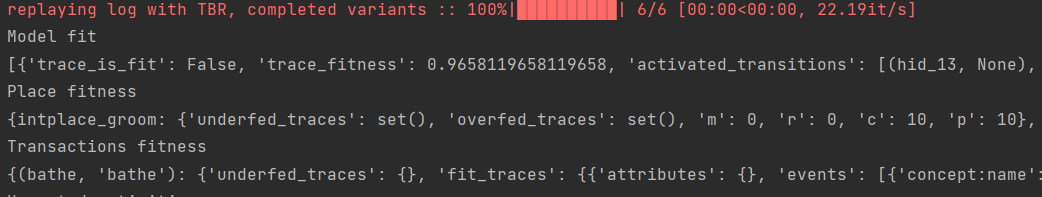


Figure 8 Example of fitness checking of the Petri Net with the log. In this case we used a heuristic miner, and then we checked the fitness, as we can see the model is not totally fit so the trace is not fit but as we can see we obtained a value near to one 1, this means that the model is quite fit with the model.

To check the function also printed the Place fitness and the Transaction fitness to check where we have some problems if it does not fit perfectly.

PM4PY, offer different simulation algorithms, that starting from a model, are able to produce an output that follows the model and the different rules that have been provided by the user.

So I also inserted the basic playout simulation of the Petri Net with Token, that tries to execute the net, to see how and if it works

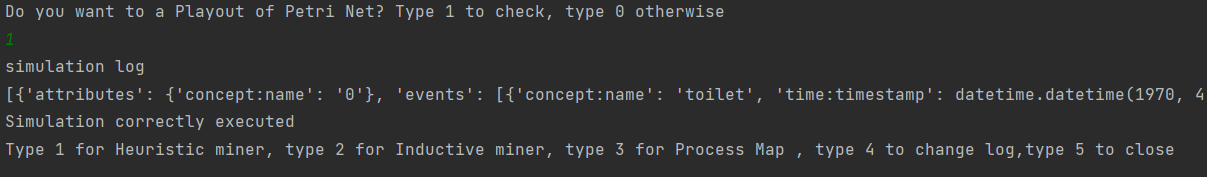


Figure 9 Simulation of a PN

We also inserted on the Inductive Mining the possibilities to see the Process Tree to see how works the reachability inside the PN generated with the Inductive Miner.

It shows us how the places are linked each one and how is possible to move inside the PN.

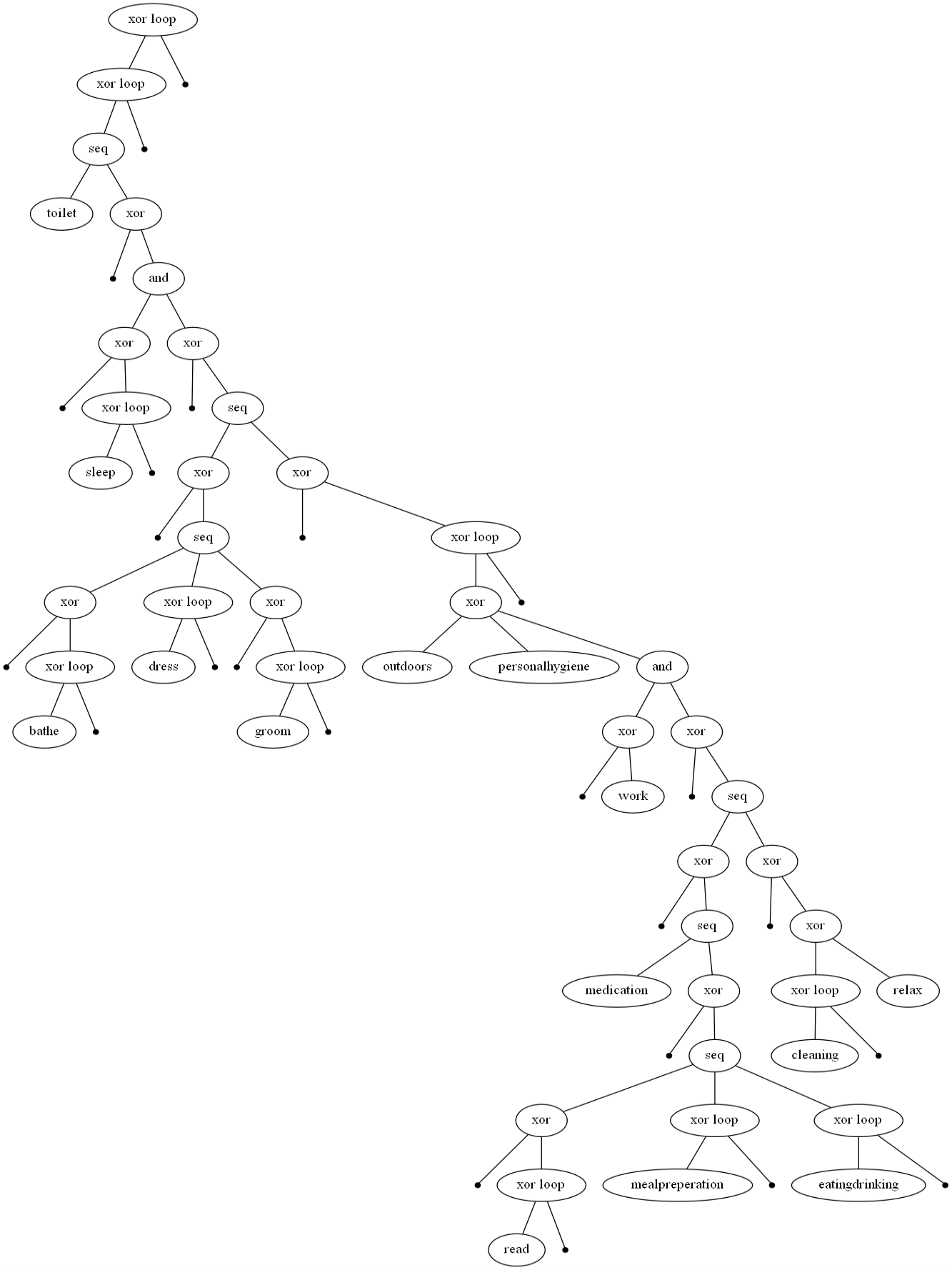


Figure 10 Example of Petri Net Process Tree generated from a Heuristic Miner.

CONCLUSIONS

At the end we simulated an execution of the created Net to see if it works and we added the possibility for the programs to save the model of the petri net generated, so it will be reusable if needed again

We can say that all the library of PM4PY and the built functions works properly, we also managed to see how all the different Mining algorithms works and the results that they give.

After the process Discovery we also managed the conformance checking and the process map that gave to us more detailed information about the built Petri Net.

Of course, this project model can be used for analyzing a log and to obtain the Petri Net of the log, but there are several more features that could be implemented. We mostly followed the documentation of PM4PY, so there are a lot of things that could be still implemented inside.

To develop this project, we first used PyCharm as IDLE, we downloaded PM4PY (Main library for PN) and Graphviz (Open-source program to draw graphs). We also imported the OS library of Python for technical reason.

About the data used in developing the Petri Net, the logs are of 4TU Research Data, these data concern the study of the sequence of the action made in a Smart Home, that with a lot of sensors, register the action of the user of the Smart Home.

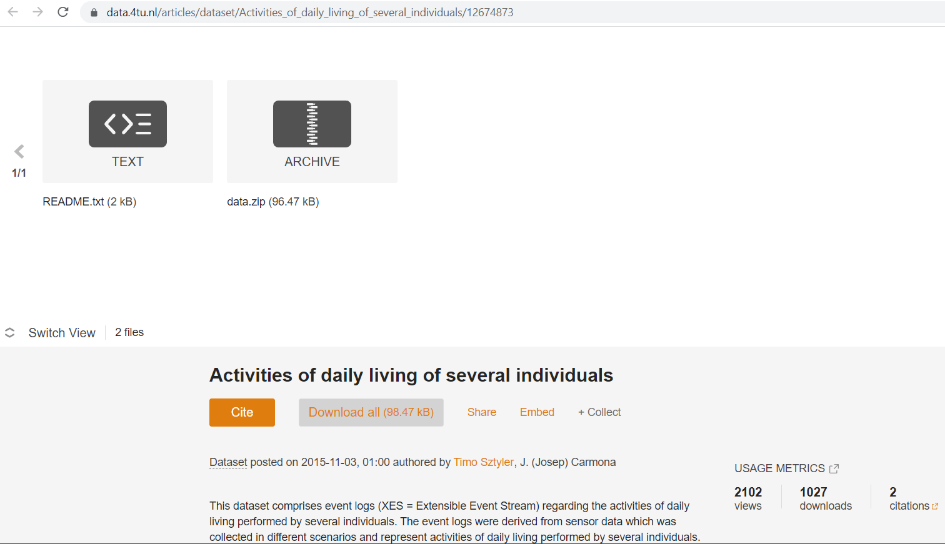


Figure 11 This is the data log that can be found at this link: <https://data.4tu.nl/articles/dataset/Activities_of_daily_living_of_several_individuals/12674873>