Comparative Analysis of Discovered Models for Processing Unit 2 (PU2)

Experimental Report

December 22, 2025

1 Introduction

This report presents the Petri net models discovered for Processing Unit 2 (PU2) of the Automated Manufacturing System.

The logic of PU2 differs fundamentally from PU1. While PU1 features a variable loop, PU2 executes a fixed sequence of operations: a part enters (e), is drilled (f), rotated (g), drilled again (f), rotated again (g), and finally exits (h).

Therefore, the ground truth is the linear sequence: $e \rightarrow f \rightarrow g \rightarrow f \rightarrow g \rightarrow h$. The main challenge for discovery algorithms here is **Cycle Sensitivity**: correctly distinguishing a fixed repetition (which should be unfolded) from a variable loop.

2 Discovered Models

2.1 Alpha Miner Family

Figure 1 displays the results from the Alpha Miner variants.

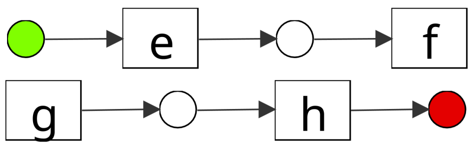
Observation: The Alpha family algorithms presented severe structural anomalies and failed to capture the sequential logic:

- **Alpha Miner** (Fig. 1a): Event g appears as an **always-enabled transition**, producing tokens for h. Event f consumes the token produced by e but does not produce any output token (acting as a sink).

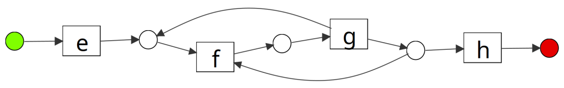
  While the sequence $e \rightarrow f \rightarrow g \rightarrow h$ can technically occur (since g can fire spontaneously after f), the **causal flow is broken**. The token representing the specific process instance initiated at e is destroyed at f. The subsequent execution of g is causally disconnected from f, meaning the model fails to represent that g should strictly follow the completion of f.

- **Alpha+ and Alpha++** (Figs. 1b and 1c): Identified the reciprocal dependency between f and g but merged the two occurrences into a single loop structure, losing the "exactly twice" constraint.

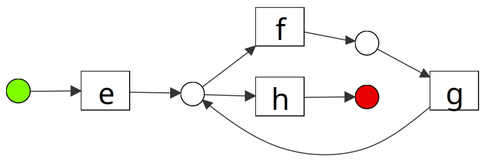
- **Alpha$** (Fig. 1d): Event g is also unrestricted but produces tokens that enable both f and h. Regarding the sequence $e \rightarrow f \ldots$: after e executes, a token is produced for f. However, f **cannot fire** immediately because it has an artificial dependency on g (it requires a token from g's output place). Thus, the standard sequence is blocked until g spontaneously fires, violating the causal order.



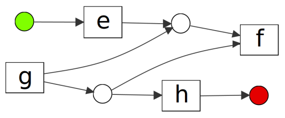
(a) Alpha Miner (Structural Break)



(b) Alpha+ Miner (Generalized Loop)



(c) Alpha++ Miner (Generalized Loop)



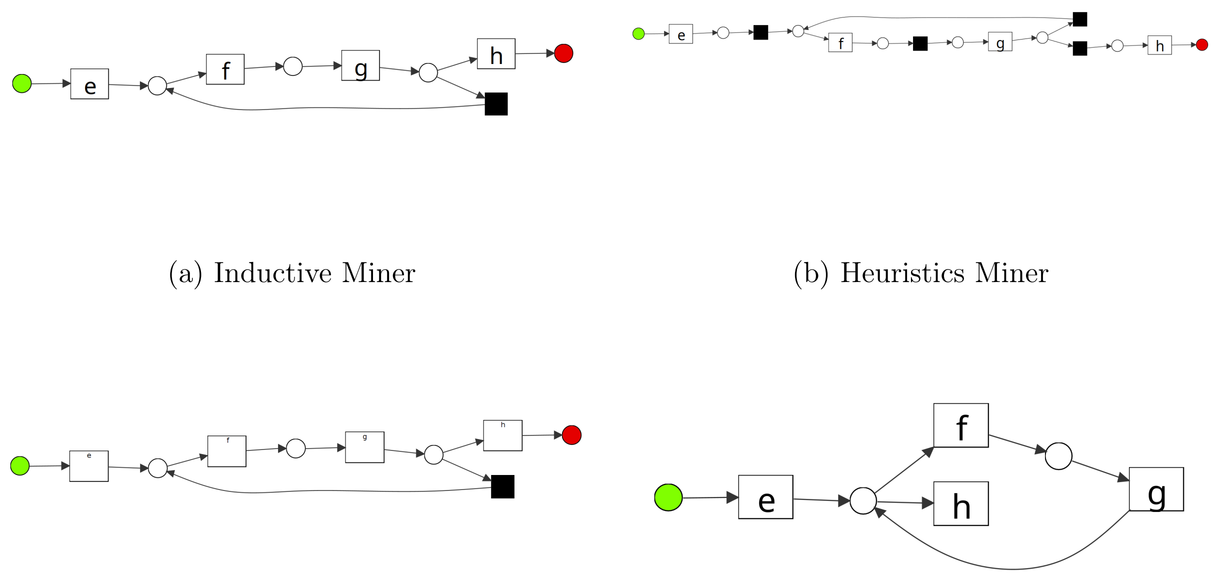
(d) Alpha$ Miner (Artificial Dependency)

Figure 1: Models discovered by the Alpha Miner family for PU2.

2.2 Heuristic, Inductive and ILP Approaches

Figure 2 groups the algorithms that typically prioritize generalization.

Observation: The **Inductive Miner**, **Heuristics Miner**, **Fodina**, and **ILP Miner** all converged to the same logical representation: a **Loop**. By identifying the repeated pattern f, g, these algorithms abstracted the fixed sequence into a cycle ($f \leftrightarrow g$). While this produces sound and compact models, it represents an **over-generalization** for this specific component, as it allows arbitrary repetitions instead of strictly two.



(a) Inductive Miner

(b) Heuristics Miner

(c) Fodina

(d) ILP Miner

Figure 2: Algorithms that generalized the sequence into a loop.

2.3 Evolutionary Tree Miner (ETMd)

Figure 3 presents the result for the ETMd algorithm.

Observation: Unlike the other methods, the **ETMd** successfully satisfied the Cycle Sensitivity criterion. It generated a model that explicitly **unfolds the sequence** $f \to g \to f \to g$. This behavior occurs because ETMd optimizes for both fitness and precision. Modeling the sequence as a loop would lower the precision score (by allowing unseen behavior). In this case, ETMd determined that the gain in precision outweighed the cost in structural complexity, resulting in the correct linear model.

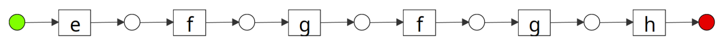


Figure 3: Evolutionary Tree Miner (ETMd) result: Correct unfolded sequence.