Process Mining and Automata Learning: evaluate controllability of extracted processes

Pietro Sala

Biomedical Decision Support Systems 24/25

1 Basic Definitions

A trace t is defined as a sequence of labels in Σ, where Σ represents the alphabet of possible event labels. Formally, a trace $t = \langle e_1, e_2, \ldots, e_n \rangle$ where each $e_i \in \Sigma$ represents an event that occurred at a specific point in the process. The length of trace t, denoted by $|t|$, is the number of events in the sequence.

Given a trace $t = \langle e_1, e_2, \ldots, e_n \rangle$ and a natural number $k > 0$, a *labelling* of t is a function ℓ_t that maps each element of the trace to a sequence of k bits. Formally, $\ell_t : \{1, 2, \ldots, |t|\} \to \{0, 1\}^k$ assigns a k-dimensional binary vector to each position in the trace.

Given a trace $t = \langle e_1, e_2, \ldots, e_n \rangle$, a *timestamp* function v_t assigns to each element of the trace a positive real number representing the time the event occurred. Formally, $v_t : \{1, 2, \ldots, |t|\} \to \mathbb{R}^+$ assigns a timestamp to each position in the trace, where $v_t(i)$ represents the time at which event e_i occurred.

A log of time-stamped labelled traces, denoted as $\mathcal{L}$, is a finite multiset of triples (t, ℓ_t, v_t) where t is a trace, ℓ_t is a labelling function for t, and v_t is a timestamp function for t. Formally, $\mathcal{L} = \{(t_1, \ell_{t_1}, v_{t_1}), (t_2, \ell_{t_2}, v_{t_2}), \ldots, (t_m, \ell_{t_m}, v_{t_m})\}$ where each triple can appear multiple times in the multiset, reflecting the frequency of that particular time-stamped labelled trace in the observed process executions.

For a log $\mathcal{L}$ of time-stamped labelled traces, we introduce the following notation:

- $\mathcal{L}(t, \ell_t, v_t)$ denotes the number of occurrences of the triple (t, ℓ_t, v_t) in $\mathcal{L}$
- $Dom(\mathcal{L})$ denotes the set of all distinct time-stamped labelled trace triples in $\mathcal{L}$
- $\mathcal{L}(t)$ represents the total number of time-stamped labelled traces in $\mathcal{L}$ with first element t, i.e., $\mathcal{L}(t) = \sum_{(\ell_t, v_t):(t,\ell_t,v_t)\in Dom(\mathcal{L})} \mathcal{L}(t, \ell_t, v_t)$
- $Traces(\mathcal{L})$ is the set of all traces appearing in $\mathcal{L}$, i.e., $Traces(\mathcal{L}) = \{t \mid \exists \ell_t, v_t : (t, \ell_t, v_t) \in Dom(\mathcal{L})\}$

2 Petri Net Extraction from Logs

A Petri Net is a mathematical modeling language for the description of distributed systems. Formally, a Petri Net is defined as a tuple $N = (P, T, F)$, where:

- P is a finite set of places
- T is a finite set of transitions with $P \cap T = \emptyset$
- $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation

Given a log $\mathcal{L}$ of time-stamped labelled traces, let $N_{\mathcal{L}} = (P, T, F, \mathcal{T})$ denote the Petri Net extracted from $\mathcal{L}$ using some process mining algorithm acting only on the traces and their count in $\mathcal{L}$ (not the labelling or timestamps), where $\mathcal{T} : T \to \Sigma \cup \{\tau\}$ is a labeling function that maps transitions to labels in $\Sigma \cup \{\tau\}$, with τ representing silent transitions.

3 Trace Alignment

An alignment between a trace $t = \langle e_1, e_2, \ldots, e_n \rangle$ and a computation of the Petri Net N (sequence of transitions) $\sigma = \langle t_1, t_2, \ldots, t_m \rangle$ is a sequence of pairs

$$\gamma = \langle (a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k) \rangle$$

where each pair (a_i, b_i) is one of:

- (e_j, t_l) with $e_j \in \Sigma$ and $t_l \in T$ such that $\mathcal{T}(t_l) = e_j$ (synchronized move)
- $(e_j, \gg)$ with $e_j \in \Sigma$ (move on trace)

- $(\gg, t_l)$ with $t_l \in T$ (move on model)

The symbol $\gg$ indicates a "skip" in either the trace or the computation. The alignment γ must preserve the order of both the trace and the computation, meaning that the non-skip elements in the first component of γ form the trace t, and the non-skip elements in the second component form the computation σ.
The cost of an alignment γ, denoted as $cost(\gamma)$, is determined by the number of misalignments, i.e., the number of pairs in which either component is a skip ($\gg$). However, moves on the model involving silent transitions (transitions labeled with τ) do not contribute to the cost, as they have no observable behavior in the process. Formally:

$$cost(\gamma) = |\{(a_i, b_i) \in \gamma : \begin{array}{c}(a_i =\gg \wedge \mathcal{T}(b_i) \neq \tau) \\ \vee (a_i \neq \gg \wedge b_i =\gg)\end{array}\}|$$

An optimal alignment between a trace t and a Petri Net N is an alignment with minimal cost.
Even though there may be multiple optimal alignments for a given trace and Petri Net, we assume the existence of a function $A_{\mathcal{L},N}$ that associates to each trace $t \in Traces(\mathcal{L})$ a single optimal alignment between t and N. That is, $A_{\mathcal{L},N} : Traces(\mathcal{L}) \to \Gamma$, where Γ is the set of all possible alignments, and for each $t \in Traces(\mathcal{L})$, $A_{\mathcal{L},N}(t)$ is an optimal alignment between t and N.

4 Construction of Aligned Traces

Given the alignment function $A_{\mathcal{L},N}$, we define the aligned log $\overline{\mathcal{L}}$ as the log obtained by transforming each trace based on its alignment. For a trace $t \in Traces(\mathcal{L})$ with alignment $A_{\mathcal{L},N}(t) = \langle (a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k) \rangle$, we construct its aligned counterpart t_A as follows:

- Events from the original trace that are synchronized with transitions (where $a_i \neq \gg$ and $b_i \neq \gg$) are kept with their original labels
- Moves on the model $(\gg, t_l)$ where $\mathcal{T}(t_l) \neq \tau$ (non-silent transitions) are converted to events with the corresponding transition label but assigned dummy labelling ϵ and dummy timestamp ϵ
- Events from the original trace that have no matching transition (where $a_i \neq \gg$ and $b_i =\gg$) are removed
- Moves on the model $(\gg, t_l)$ where $\mathcal{T}(t_l) = \tau$ (silent transitions) are not included in the aligned trace

For the labelling and timestamping of the aligned log, we define ℓ_{t_A} and v_{t_A} as follows:

- If an event is kept from the original trace (first case above), we use the original labelling ℓ_t and the original timestamp v_t
- If an event is introduced by the model (second case above), we assign it the dummy labelling ϵ and the dummy timestamp ϵ

Formally, the aligned log $\overline{\mathcal{L}}$ consists of triples $(t_A, \ell_{t_A}, v_{t_A})$ with the same multiplicity as the corresponding (t, ℓ_t, v_t) in $\mathcal{L}$.
It is important to note that by construction, while the aligned trace t_A may be shorter than the corresponding Petri Net computation (due to the omission of silent transitions), the alignment between t_A and the Petri Net N is perfect in terms of observable behavior. That is, for every trace t_A in $\overline{\mathcal{L}}$, each event in t_A corresponds exactly to a transition in N with the same label, and any silent transitions in N are represented as $(\gg, t_l)$ pairs in the alignment where $\mathcal{T}(t_l) = \tau$.

5 Alignment Between Aligned Log and Petri Net

Let $\overline{A}$ denote the alignment between the aligned log $\overline{\mathcal{L}}$ and the Petri Net N. For each trace t_A in the aligned log $\overline{\mathcal{L}}$, we have $\overline{A}(t_A)$ as the alignment between t_A and a computation of N.
By construction, for every trace $t_A = \langle e_1, e_2, \ldots, e_k \rangle$ in $\overline{\mathcal{L}}$, the alignment $\overline{A}(t_A)$ is a sequence of pairs:

$$\overline{A}(t_A) = \langle (a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m) \rangle$$

where each pair (a_i, b_i) is one of:

- (e_j, t_l) with $e_j \in \Sigma$ and $t_l \in T$ such that $\mathcal{T}(t_l) = e_j$ (synchronized move)
- $(\gg, t_l)$ with $t_l \in T$ such that $\mathcal{T}(t_l) = \tau$ (silent move on model)

Note that in the aligned log, there are no moves on trace $(e_j, \gg)$ or non-silent moves on model $(\gg, t_l)$ where $\mathcal{T}(t_l) \neq \tau$, as these have been resolved during the alignment process. The alignment $\overline{A}(t_A)$

includes all synchronized moves between the trace events and transitions, as well as any silent transitions in the Petri Net that are part of the computation. The cost of this alignment is zero since silent transitions do not contribute to the cost.
Given an alignment $\gamma = \langle (a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m) \rangle$ and a position i in γ, we define the *sample* of such a pair as $z = (x, y)$ where:

- x is the sequence of events from trace t_A up to (but excluding) the position i, after removing all skips $\gg$ and events with dummy labelling ϵ (i.e., $\ell_{t_A}(a_i) = \epsilon$). This ensures that only actual observed events (not those introduced during alignment) are considered in the sample. Formally, $x = \langle a_j \mid 1 \leq j < i, \ell_{t_A}(a_i) \neq \epsilon, a_j \neq \gg \rangle$.
- y is the transition at position i in the alignment, i.e., $y = b_i$.

Intuitively, a sample (x, y) captures the relationship between a prefix of actual observed events in the trace and the next transition that follows in the Petri Net computation. This provides a view of how a sequence of past events relates to the subsequent step in the process execution. We call it $z_{t_A, \ell_A, v_{t_A}}$, let us notice that due to the labelings the same alignments may have different samples.

6 States of the Process

A state Q is defined as a partial function mapping each element of the aligned log $\overline{\mathcal{L}}$ to an index in the perfect alignment between that trace and the Petri Net. Formally, $Q : \overline{\mathcal{L}} \rightharpoonup \mathbb{N}$ is a partial function such that:
There exists a unique marking M_Q of the Petri Net N such that for each trace $t_A \in Dom(Q)$, the computation of N (from the initial marking) up to (but excluding) the transition at position $Q(t_A)$ in the alignment $\overline{A}(t_A)$ reaches exactly this marking M_Q.
In other words, a state Q identifies a specific position in the perfect alignment of each trace in its domain, such that all these positions correspond to the same marking in the Petri Net. This allows for the inclusion of silent transitions in the Petri Net, which appear in the alignment but not in the trace itself.
The domain of Q, denoted $Dom(Q)$, is the subset of traces from the aligned log $\overline{\mathcal{L}}$ to which Q assigns an alignment index.
The initial state, denoted as Q_0, is a special state that maps each trace in the aligned log to index 1 in its perfect alignment, representing the position before the first transition in the computation. Formally, for each trace $t_A \in Traces(\overline{\mathcal{L}})$:

$$Q_0(t_A) = 1$$

The initial state represents the starting point of all process executions according to the Petri Net model, where the net is in its initial marking before any transitions have been executed.

7 Transitions Between States and Markov Process

Let Q and Q' be two states as defined previously. We say that there is a *transition via* t between Q and Q', denoted $Q \xrightarrow{t} Q'$, if and only if:

- Q' is defined only for those traces $tr \in Dom(Q)$ such that at the $Q(tr)$, the alignment $\overline{A}(tr)$ points to a transition t and $Q'(tr) = Q(tr) + 1$.

In other words, Q' captures the state of the process after executing transition t from state Q, considering only those traces that can actually execute t from the current position.
Furthermore, for a given state Q with its associated unique marking M_Q, let $Enabled(M_Q) = \{t_1, t_2, \ldots, t_k\}$ be the set of all transitions enabled in marking M_Q. For each enabled transition $t_i \in Enabled(M_Q)$, let Q_i be the state such that $Q \xrightarrow{t_i} Q_i$.
We define the probability of transitioning from state Q to state Q_i via transition t_i as:

$$P(Q \xrightarrow{t_i} Q_i) = \frac{|Dom(Q_i)|}{|Dom(Q)|}$$

This ratio represents the proportion of traces in the domain of Q that follow transition t_i. By construction, these probabilities form a valid probability distribution over all enabled transitions:

$$\sum_{t_i \in Enabled(M_Q)} P(Q \xrightarrow{t_i} Q_i) = 1$$

Therefore, the transition relation between states defines a Markov process with Q_0 as initial state.

Assignment 1: Extracting the Markov Process from a Log

Students are tasked with extracting the Markov process associated with a given log by implementing the following steps:

1. Extract a Petri Net from the provided log using an appropriate process mining algorithm.
2. Perform alignment between the log and the extracted Petri Net to obtain an aligned log and the relative perfect alignment.
3. Build an automata learning wrapper with the following specifications:
    - **Initialization and Reset:**
        - Select a current alignment randomly from the perfect alignment
        - Initialize the current marking of the net to its initial marking
        - Set the current index to 1
    - **Step Function:** The step function (with a single alphabet of input) should:
        - Return a pair consisting of:
            * The transition at the current index
            * The marking of the Petri Net after executing this transition (which will be saved as the new current marking)
        - Increment by 1 the current index
        - If the index is out of bounds (meaning the net has reached its final marking), return only the final marking

8 Entropy of a Process

The entropy of a state Q in a Markov process provides a measure of uncertainty or randomness in the next transition from that state. Formally, the entropy of state Q, denoted $H(Q)$, is defined as:

$$H(Q) = -\sum_{t_i \in Enabled(M_Q)} P(Q \xrightarrow{t_i} Q_i) \log_2 P(Q \xrightarrow{t_i} Q_i)$$

where:

- $Enabled(M_Q)$ is the set of transitions enabled in the marking M_Q associated with state Q
- $P(Q \xrightarrow{t_i} Q_i)$ is the probability of transitioning from state Q to state Q_i via transition t_i, defined as $\frac{|Dom(Q_i)|}{|Dom(Q)|}$

The entropy value has the following interpretations:

- A low entropy value (close to 0) indicates that the process behavior at state Q is highly predictable, with one transition being significantly more likely than others
- A high entropy value indicates that the process behavior at state Q is more random or unpredictable, with multiple possible transitions having similar probabilities
- The maximum entropy for a state with n enabled transitions is $\log_2(n)$, which occurs when all transitions are equally likely

The entropy of states in a process model provides valuable insights into decision points, process variability, and potential areas for process improvement or control.
Building upon the entropy of individual states, we can define the entropy of an entire process as the weighted sum of the entropies of all states in the Markov process. Formally, the entropy of a process, denoted $H(Process)$, is defined as:

$$H(Process) = \sum_{Q \in States} P(Q) \cdot H(Q)$$

where:

- $States$ is the set of all states in the Markov process
- $P(Q)$ is the probability of reaching state Q in the process from the initial state I'll refine the definition to incorporate the counting of identical traces in the aligned log.

  The probability of reaching a state Q in our Markov process, denoted as $P(Q)$, can be computed as the proportion of alignment-index pairs in the entire aligned log for which state Q is reached, accounting for the multiplicities of traces. Formally:

$$P(Q) = \frac{\sum_{t_A \in Dom(Q)} \overline{\mathcal{L}}(t_A)}{\sum_{t_A \in Traces(\overline{\mathcal{L}})} \overline{\mathcal{L}}(t_A) \cdot |t_A|}$$

where:

- $Dom(Q)$ is the set of traces from the aligned log mapped by state Q to specific indices in their alignments
- $\overline{\mathcal{L}}(t_A)$ represents the number of occurrences (multiplicity) of trace t_A in the aligned log $\overline{\mathcal{L}}$
- $|t_A|$ is the length of trace t_A
- The denominator represents the total number of positions across all traces in the aligned log, accounting for their multiplicities

This definition ensures that the probability of a state is proportional to the frequency with which that state is visited during process executions recorded in the log, providing a statistically sound basis for computing the entropy of the overall process.

- $H(Q)$ is the entropy of state Q as defined previously

The probability $P(Q)$ of reaching a state Q can be computed from the Markov process by analyzing the stationary distribution of the process or by counting the frequency of state visits in the observed traces.
This process-level entropy provides a comprehensive measure of the overall predictability or randomness of the process. A low process entropy indicates a highly structured and predictable process, while a high process entropy suggests a process with significant variability and numerous decision points.

9 Literals and Local Formulas

A literal is either p_h where h ranges among $1...k$ or its negation $\neg p_h$. A local formula is any combination of conjunction and negation of literals.
A labelling of an event satisfies a literal p_h if and only if the h-th component of its k-vector is 1. Similarly, a labelling satisfies a literal $\neg p_h$ if and only if the h-th component of its k-vector is 0.
More formally, given a trace $t = \langle e_1, e_2, \ldots, e_n \rangle$ with labelling function $\ell_t : \{1, 2, \ldots, |t|\} \to \{0, 1\}^k$, and considering position i in the trace:

- $\ell_t(i)$ satisfies literal p_h if and only if the h-th component of $\ell_t(i)$ is 1
- $\ell_t(i)$ satisfies literal $\neg p_h$ if and only if the h-th component of $\ell_t(i)$ is 0
- $\ell_t(i)$ satisfies a conjunction of literals if and only if it satisfies each literal in the conjunction
- $\ell_t(i)$ satisfies a negation of a formula if and only if it does not satisfy the formula

A labelling of an event satisfies a local formula if the evaluation of the formula using the corresponding k-vector results is true.

10 Dataset Partitioning Based on Trace Tests

A trace test is any formula of the form (ψ, ts) where ψ is a local formula and ts is a non-negative real number.
Given a trace sample $x = \langle e_1, e_2, \ldots, e_n \rangle$, ℓ, v, a trace test (ψ, ts) holds on x if there exists an event e_i in x such that:

- e_i occurs at most ts time before the last event e_n of x (i.e., $v(n) - v(i) \leq ts$)
- The labelling $\ell(i)$ satisfies the local formula ψ

Note that the last event e_n of x is always included in the check since ts is at least 0, meaning that $v(n) - v(n) = 0 \leq ts$.
Given a state Q, we can associate a dataset $Z_Q = (X_Q, Y_Q)$ consisting of samples obtained from traces in $Dom(Q)$ derived at index Q.
The dataset Z_Q consists of all such samples derived from traces in $Dom(Q)$:

$$Z_Q = \left\{ (x, y) \mid \begin{array}{c} t_A \in Dom(Q), (x, y) \text{ is the} \\ \text{sample for } t_A \text{ at index} \\ Q(t_A) \text{ according to } \ell_{t_a}, v_{t_a} \end{array} \right\}$$

This dataset captures the relationship between the observed event sequences and the subsequent transitions at state Q, providing the empirical basis for learning predictive models of process behavior at this particular state.
Given a dataset Z_Q associated with state Q and a trace test (ψ, ts), we can partition Z_Q into two complementary subsets: $Z_Q^{\top}$ for samples that satisfy the trace test, and $Z_Q^{\perp}$ for samples that do not satisfy it, formally:

- $Z_Q^\top = \{(x, y) \in Z_Q \mid$ trace test (ψ, ts) holds on $x\}$
- $Z_Q^\perp = \{(x, y) \in Z_Q \mid$ trace test (ψ, ts) does not hold on $x\}$

By construction, $Z_Q^\top \cup Z_Q^\perp = Z_Q$ and $Z_Q^\top \cap Z_Q^\perp = \emptyset$, forming a complete partition of the original dataset.

This partitioning enables the analysis of how specific temporal and logical conditions (represented by the trace test) affect the subsequent process behavior, potentially revealing patterns and correlations between historical events satisfying certain properties and future process steps.

Let us notice that the entropy of the random variable Y_Q associated with the dataset $Z_Q = (X_Q, Y_Q)$ is exactly the entropy of state Q.

Given a trace test (ψ, ts) that partitions each state dataset Z_Q into $Z_Q^\top$ and $Z_Q^\perp$, the entropy of this trace test over the entire process is calculated by weighting the entropies of these partitions by both their relative sizes and the probability of reaching each state.

Mathematically, the entropy of a trace test over a process can be expressed as:

$$H(\text{TraceTest}) = \sum_{Q \in \text{States}} P(Q) \cdot \left(\frac{|Z_Q^\top|}{|Z_Q|} \cdot H(Y_Q^\top) + \frac{|Z_Q^\perp|}{|Z_Q|} \cdot H(Y_Q^\perp) \right)$$

Where: - $P(Q)$ is the probability of reaching state Q in the Markov process - $\frac{|Z_Q^\top|}{|Z_Q|}$ is the proportion of samples in state Q that satisfy the trace test - $\frac{|Z_Q^\perp|}{|Z_Q|}$ is the proportion of samples in state Q that do not satisfy the trace test - $H(Y_Q^\top)$ is the entropy of the transition distribution for samples satisfying the trace test - $H(Y_Q^\perp)$ is the entropy of the transition distribution for samples not satisfying the trace test

This formula captures how the trace test affects the predictability of the process across all states, weighted by the importance (probability) of each state in the overall process.

For each state, we calculate the weighted average of the entropies of the two partitions created by the trace test, and then sum these weighted averages across all states, weighting by the probability of reaching each state.

Then the Information Gain for trace test is:

$$IG = H(\text{Process}) - H(\text{TraceTest})$$

Assignment 2: Calculating Entropy and Information Gain in Process Mining

Students are tasked with calculating entropy measures and information gain for process models extracted from event logs, building upon the Markov process constructed in Assignment 1:

1. Calculate the entropy of each state Q in the extracted Markov process using the formula:

$$H(Q) = -\sum_{t_i \in Enabled(M_Q)} P(Q \xrightarrow{t_i} Q_i) \log_2 P(Q \xrightarrow{t_i} Q_i)$$

2. Calculate the overall entropy of the process as a weighted sum:

$$H(Process) = \sum_{Q \in States} P(Q) \cdot H(Q)$$

where $P(Q)$ is the probability of reaching state Q.

3. For a given trace test (ψ, ts), calculate:

- The partitioned datasets $Z_Q^\top$ and $Z_Q^\perp$ for each state Q
- The entropy of each partition $H(Y_Q^\top)$ and $H(Y_Q^\perp)$
- The weighted trace test entropy: $H(TraceTest)$
- The information gain: $IG = H(Process) - H(TraceTest)$

11 Trace Test Relabeling

Given a set of trace tests $TTS = \{(\psi_1, d_1), (\psi_2, d_2), ..., (\psi_k, d_k)\}$ and a log $\mathcal{L}$, the relabeling of $\mathcal{L}$ with TTS is a transformation of each trace $t = \langle e_1, e_2, ..., e_n \rangle$ in the log.

In this transformation, each event e_i in the original trace is replaced by a new event e_i' that is augmented with a binary vector in $\{0, 1\}^k$. Formally, we define:

$$e_i' = (e_i, tt_i)$$

where $tt_i \in \{0, 1\}^k$ is a binary vector such that for each $j \in \{1, 2, ..., k\}$:

$$tt_i[j] = \begin{cases} 1 & \text{if trace test } (\psi_j, d_j) \text{ holds at position } i \\ 0 & \text{otherwise} \end{cases}$$

A trace test (ψ_j, d_j) holds at position i if there exists a position $r \leq i$ such that: 1. The time difference between events at positions r and i does not exceed d_j: $v_t(i) - v_t(r) \leq d_j$ 2. The labeling at position r satisfies the local formula ψ_j: $\ell_t(r)$ satisfies ψ_j

The resulting relabeled log $\mathcal{L}'$ consists of traces $t' = \langle e'_1, e'_2, ..., e'_n \rangle$ where each event is enhanced with the information about which trace tests hold at that position. The original labeling function ℓ_t and timestamp function v_t remain unchanged.

Assignment 3: Trace Test Relabeling and Entropy Analysis

Given a set of trace tests $TTS = \{(\psi_1, d_1), (\psi_2, d_2), ..., (\psi_k, d_k)\}$ and a log $\mathcal{L}$, the relabeling of $\mathcal{L}$ with TTS is defined as follows:

For each trace $t = \langle e_1, e_2, ..., e_n \rangle$ in the log with labeling function ℓ_t and timestamp function v_t:

1. Transform each event e_i into a new event $e'_i = (e_i, tt_i)$ where $tt_i \in \{0, 1\}^k$ is a binary vector such that for each $j \in \{1, 2, ..., k\}$:

$$tt_i[j] = \begin{cases} 1 & \text{if trace test } (\psi_j, d_j) \text{ holds at position } i \\ 0 & \text{otherwise} \end{cases}$$

2. A trace test (ψ_j, d_j) holds at position i if there exists a position $r \leq i$ such that: - $v_t(i) - v_t(r) \leq d_j$ (time constraint) - $\ell_t(r)$ satisfies ψ_j (logical constraint)

3. The resulting relabeled log $\mathcal{L}'$ consists of traces $t' = \langle e'_1, e'_2, ..., e'_n \rangle$ where each event is enhanced with the information about which trace tests hold at that position.

For this assignment:

- Select at least two different sets of trace tests: - Set A: A single trace test (ψ_A, d_A) - Set B: Multiple trace tests $\{(\psi_1, d_1), (\psi_2, d_2), ..., (\psi_k, d_k)\}$
- Apply the relabeling procedure to create two new logs $\mathcal{L}_A$ and $\mathcal{L}_B$
- For each log (original and relabeled):
  - Extract the Petri Net and corresponding Markov process
  - Calculate the process entropy using the methods from Assignment 2
  - Compare the entropy values to analyze how trace test relabeling affects process predictability
- Analyze which trace tests or combinations of trace tests provide the most significant entropy reduction and explain the implications for process controllability.