0.0.1 Process Logs, Cases and Instance Sequences

We start by formalising the log and its elements. Let $\mathcal{E}$ be the universe of event identifiers and $E \subseteq \mathcal{E}$ a set of events. Let C be a set of case identifiers and $\pi_\sigma : E \mapsto C$ a surjective function that links every element in E to a case $c \in C$ in which c signifies a case. For a set of events $E \subseteq \mathcal{E}$, we use a shorthand $s \in C$ being a particular sequence $s^c = \langle e_1, e_2, \ldots, e_t \rangle$ as a case c of length t.

Furthermore, let $\mathcal{T}$ be the time domain and $\pi_t : E \mapsto \mathcal{T}$ a surjective linking function which strictly orders a set of events.

Let $\mathcal{A}$ be a fixed collection of named attribute sets of size I and $A_i \in \mathcal{A}$ be a specific named attibute set.

Then, each e_t consists of a set $e_t = \{a_1 \in A_1, a_2 \in A_2, \ldots, a_i \in A_i\}$ with the size I, in which each a_i refers to a value within its respective named attribute set. Conversely, $\pi_A : a_i \mapsto A_i$ maps any attribute a_i value to their respective named attribute set.

Furthermore, let $\pi_d : A_i \mapsto \mathbb{N}$ be a surjective function, which determines the dimensionality of a_i and also F be a set of size I containing a representation function for every named attribute set. We denote each function as a mapper to a vector space $f_i : a_i \mapsto R_i^d$, in which d represents the dimensionality of an attribute value $d = \pi_d(A_i)$.

With these definitions, we denote any event $e_t \in s^c$ of a specific case c as a vector, which concatenates every attribute representation f_i as $\mathbf{e}_t^c = [f_1; f_2; \ldots; f_j]$. Furthermore, if we refer to a specific named attribute set A_i as a name, we will use the shorthand $\overline{a}_i$.

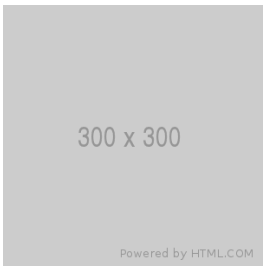

Figure 1: This figure show the representation of a particular event e and a case c.

0.0.2 State-Space Models

Generally speaking, every time-series can be represented as a state-space model[1]. Within this framework the system consists of *input states* for

subsequent states and *subsequent outputs*. A mathematical form of such a system is shown in Equation 1.

$$\begin{aligned} \mathbf{z}_{t+1} &= h(t, \mathbf{z}_t, \mathbf{u}_t) \\ \mathbf{e}_t &= g(t, \mathbf{z}_t, \mathbf{u}_t) \\ \mathbf{z}_{t+1} &:= \frac{d}{dt}\mathbf{z}_t \end{aligned} \tag{1}$$

Here, $\mathbf{u}_t$ represents the input, $\mathbf{z}_t$ the state at time t. The function h maps t, $\mathbf{z}_t$ and $\mathbf{u}_t$ to the next state $\mathbf{z}_{t+1}$. The event $\mathbf{e}_t$ acts as an output computed by function g which takes the same input as h. The variables $\mathbf{z}_t$, $\mathbf{u}_t$ and $\mathbf{e}_t$ are vectors with discrete or continuous features. The distinction of $\mathbf{z}_{t+1}$ and $\mathbf{e}_t$ decouples *hidden*[1] states, from *observable* system outputs. Figure 2 shows a graphical representation of these equations.

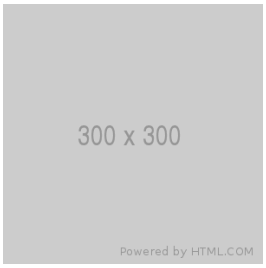

Figure 2: This figure shows a simplified graphical representation of a state-space model. Each arrow represents the flow of information.

The body of literature for state-space models is too vast to discuss them in detail[2]. However, for process mining, we can use this representation to discuss the necessary assumptions for process mining. In line with the process-definition in **??**, we can understand the event log as a collection of the observable outputs of a state-space model. The state of the process is hidden as the *true* process which generated the data cannot be observed as well. The time t is a step within the process. Hence, we will treat t as a discrete scalar value to denote discrete sequential time steps. Hence, if we have $\sigma = \{a, b, b, c\}$, then t, describes the index of each element in σ. The input $\mathbf{u}_t$ represents all context information of the process. Here, $\mathbf{u}_t$ subsumes observable information such as the starting point and process instance-related

[1] A state does not have to be hidden. Especially, if we know the process and the transition rules. However, those are often inaccessible if we only use log data. Instead, many techniques try to approximate the hidden state given the data instead.

[2] For an introduction to state-space models see: XXX

features. The functions h and g determine the transition of a process' state to another state and its output over time. **Note, that this formulation disregards any effects of future timesteps on the current timestep. Meaning, that the state transitions are causal.** As we establish in **??**, we can assume that a process is a discrete sequence, whose transitions are time-variant. In this framework, we try to identify the parameters of the functions h and g. Knowing the functions, it becomes simple to infer viable counterfactuals. However, the function parameters are often unknown and therefore, we require probablistic approaches.

We can formulate Equation 1 probablistically as shown in Equation 2.

$$\mathbb{E}[p\left(z_{t+1} \mid t, z_{1:T}, u_{1:T}, x_{1:T}, \theta_h\right)] = \int z_{t+1} \cdot p\left(z_{t+1} \mid t, z_{1:T}, u_{1:T}, x_{1:T}, \theta_h\right) \quad (2)$$

$$\mathbb{E}[p\left(x_t \mid t, z_{1:T}, u_{1:T}, \theta_g\right)] = \int x_t \cdot p\left(x_t \mid t, z_{1:T}, u_{1:T}, \theta_g\right)$$

Note, that h and g are substitued with probability density functions parametrized with θ_h and θ_g. T signifies the full sequence including future timesteps. Both expectations are intractable as they require integrating over n-dimensional vectors. To solve the intractability, we characterize the system as a *Hidden Markov Process* and Probalistic Graphical Model (PGM). This framework allows us to leverage simplifying assumptions such as the independece from future values and *d-seperation*. The stochastic process is shown in Figure 3.

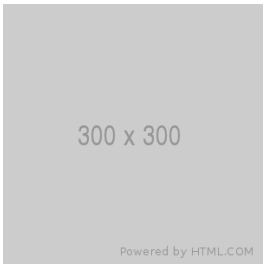

Figure 3: Figure shows a graphical representation of the stochastic process.

These characteristics change the probabilities in Equation 2 to Equation 3:

$$p\left(z_{t+1} \mid z_{1:t}, u_{1:t}, \theta_h\right) = \prod_{1}^{t} p\left(z_t \mid z_{1:t}, u_t, \theta_h\right) \tag{3}$$

$$p\left(x_t \mid z_{1:t}, \theta_g\right) = \prod_{1}^{t} p\left(x_{t-1} \mid z_{1:t}, \theta_f\right) \tag{4}$$

For $p\left(z_{t+1} \mid t, z_{1:T}, u_{1:T}, x_{1:T}, \theta_h\right)$, we ignore future timesteps, as T changes into t. *d-seperation* allows us to ignore all $\mathbf{e}_t$ of previous timesteps. The graphical form also decomposes the probability into a product of probabilities that each depend on all previous states and its current inputs. Previous $\mathbf{e}_t$ are ignored due to *d-seperation*. $p\left(x_t \mid t, z_{1:T}, u_{1:T}, \theta_g\right)$ only depends on its current state, which is in line with Hidden Markov Models (HMMs). Note, that we deliberately not assume a *strong Markov Property*, as the Deep Learning-Framework allows us to take all previous states into account. The *strong Markov Property* would assume that only the previous state suffices. At last, we assume that we do not model automatic or any other process whose state changes without a change in the input or previous states. Hence, we remove the dependency on the independet t variable. Only the previous states $z_{1:T}$ and the input information $\mathbf{u}_t$ remain time-dependent.

In this probablistic setting, the generation of counterfactuals, amounts to drawing samples from the likelihood of Equation 3. We then use the samples to reconstruct the most-likely a counterfactual $e^*_{1:t}$. Hence, our goal is to maximize both likelihoods.

[A number of AI techniques where developed to model this representation bla bla bla (HMM, Kalman, etc – Has further formalisation).]