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 $\sigma \in C$ being a particular sequence $\sigma^c = \langle e_1, e_2, \ldots, e_t \rangle$ as a case 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.

We assume that every e contains a set of data attributes. Hence, let $A = a_1, \ldots, a_i$ be a set of attribute names (e.g., timestamp, resource, action, etc.) and $\mathcal{F}$ be the universe of each attribute's representations $F \in \mathcal{F}$. For each element in A, we have a surjective function $\pi_{a_i} : A \mapsto F_i$ which links each event attribute a_i to its value $f_i \in F_i$. Each attribute value is represented with a vector space of varying dimensions $f_i \in R^d$.

We represent each event as a concatenated vector of its attribute values as $e_t^c = [f_1^c; f_2^c; \ldots; f_i^c]$, in which c specifies a particular case, a specific event within the case at time t and i its event attributes.

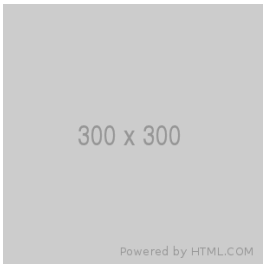

Figure 1: This figure show the representation of a particular event e and a case c.

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.

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

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

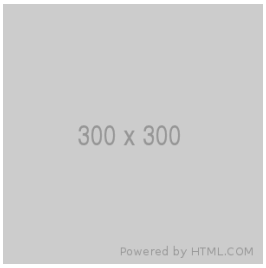

Figure 2: This figure shows a simplified graphical representation of a state-space model. Each arrow represents the flow of information.

The body of literature for state-space models is too vast to discuss it in detail[2]. However, for process mining we can use this representation to discuss the necessary assumptions with regards to process mining. In line with the process-definition in **??**, we can understand the Event Log as a collection of 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 skalar value to denote discrete sequential time steps. The input u represents all context information of the process. Here, u subsumes observable information such as the starting point and Process Instance-related 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**

[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

disregards any effects of future timesteps on the current timestep. As we establish in **??**, we can assume that a process is a discrete sequence, whose transitions are time-variant. In this framework, the goal becomes to identify the parameters of the functions h and g. Knowing the functions it is simple to infer valid counterfactuals. However, the function parameters are often unknown and therefore, 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.

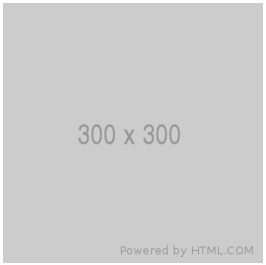

Figure 3: Figure shows a graphical representation of the stochastic process.

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

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

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

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 x 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 x_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 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 $x^*_{1:t}$. 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).]