Dynamic Bayesian Networks (DBN) – a Bayesian network that represents a temporal probability model by having state variables $\mathbf{X_t}$ replicated over time slices with the same conditional independences. We also have evidence at each time slice $\mathbf{E_t}$. For simplicity we assume a 1st order Markov process → a node's parents are either in the current or previous time slice.

- DBNs take advantage of the sparseness of the temporal probability model, whereas the equivalent HMM assumes all internal state is dependent.
- DBNs can model arbitrary distributions (thus extending beyond the capabilities of a Kalman Filter) allowing it to capture nonlinearities other models cannot.
- *Constructing DBNs*
  - We need 3 broad types of information:
    1. a prior distribution on the initial variables: $\mathbf{P}(\mathbf{X_0})$
    2. a transition model: $\mathbf{P}(\mathbf{X_{t+1}} \mid \mathbf{X_t})$
    3. a sensor model: $\mathbf{P}(\mathbf{E_t} \mid \mathbf{X_t})$
  - In addition, we must specify a local and temporal topology of the nodes at the current state and the nodes at the previous state.
  - Since the transition & sensor models are assumed to be stationary they remain the same over time → *only need to specify for initial time slice.*
  - Issues we need to deal with:
    - *Noise*: we assume that our measurements are noisy, which we model with a **Gaussian error model**.
    - *Failure*: in the real-world, sensors fail – we need to model this effect.
      - *In order to properly handle sensor failure, the sensor model must explicitly include the possibility of failure.*
      - ***transient failure model*** – allocates a probability that the sensor will return some nonsense value. This has the effect of "*inertia*" to prevent radical shifts due to intermediate failures.
      - ***persistent failure model*** **–** describes how a sensor behaves under normal and failure conditions. In particular, we have a small probability of failure, but it also models the fact that sensors tend to remain broken.

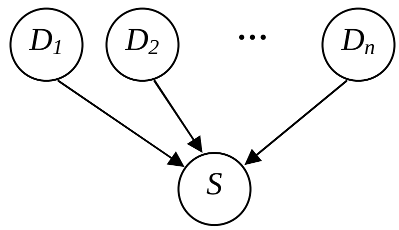
- *Exact Inference* – given a sequence of *n* observations, we simply construct the necessary DBN of *n* time slices – a process known as ***unrolling***.
    - But naively constructing the unrolled network requires *O(t)* space and inference at each time step increases at *O(t)*.
    - A more efficient process uses ***variable elimination*** before proceeding to the next time slice – this is equivalent to starting at $\mathbf{X_t}$ with a new initial distribution determined by our variable elimination.
        - This process exactly mimics the operation of a recursive filtering update. This allows us to have constant space and time per slice.
        - *Unfortunately*, the constant is exponential in number of state variables.
        - *We cannot efficiently and exactly reason about the complex temporal processes represented by general DBNs.*
- *Approximate Inference* – to estimate inference on a DBN we need to overcome a few obstacles:
    - Overcoming these blocks relies on 2 observations.
        - Again, unrolling the network is inefficient. Again, we run the samples through the network one slice at a time. *We use the samples as approximate representations of the current state distribution*.
        - Generating the samples with naïve likelihood weighting will have ~0 probability of matching the evidence. Thus, w.h.p. the samples will be independent of the evidence and will have no weight.
            - Thus, we require exponential samples to get accuracy.
            - *Instead, we want to focus the set of samples on the high-probability regions of the sate space.* We simply throw out samples of very low weight.
    - **<u>particle filtering</u>** – leverages the above observations to make an efficient sampling algorithm that is ***consistent***. We begin with *N* samples from the prior distribution at time 0: $\mathbf{P}(\mathbf{X_0})$. Then we use an <u>update cycle</u>:
        - Each sample is propagated to next time slice by sampling the next state value $\mathbf{x_{t+1}}$ given $\mathbf{x_t}$ using the transition model $\mathbf{P}(\mathbf{X_{t+1}} \mid \mathbf{X_t})$.
        - Each sample is weighted by the likelihood it assigns to the new evidence: $\mathbf{P}(\mathbf{e_{t+1}} \mid \mathbf{x_{t+1}})$ from the sensor model.
        - A new population of *N* samples is *resampled*: each new sample is selected proportional to its likelihood weight.

Noisy-OR

Suppose there are n diseases D_i all of which cause a symptom S. In the classical logic world, we might think that if you at least one of the diseases D_i than you would have symptom S and you wouldn't have it otherwise. This is modeled by the following logical sentence (a simple OR-gate):

$$S = D_1 \vee D_2 \vee \ldots \vee D_n$$

Of course, we want to incorporate uncertainty into the picture. This is captured by a particular model known as the *Noisy-OR* model. The general graphical structure for this model is simply:



However, this graphical structure does not capture all the intricacies we specified in the logical setting (In fact, the above graphical model is the same for *Noisy-AND* and many other "Noisy" versions of logical gates). The concept of *Noisy-OR* must be captured in the conditional probability table. It must have the following properties:

1. We want to model the probabilistic structure of OR such that (roughly) $S=true$ if any one of the diseases is present and $S=false$ otherwise.
    a. $P\left(S = true \mid D_1 = D_2 = \ldots = D_n = false\right) = 0$
    b. $P\left(S = false \mid D_1 = D_2 = \ldots = D_n = false\right) = 1$
2. It seems bad form to say there is 0 probability of having a symptom… couldn't there be causes we're not accounting for?
    a. *We assume we've accounted for ALL causes*. Any miscellaneous causes can be captured by an extra **leak node**.
3. Even if a cause (disease) is present, the effect (symptom) might be inhibited. This is the uncertainty we wish to model.
    a. Each cause can be inhibited with probability q_i. Thus,
    $$P\left(S = false \mid D_1 = D_2 = \ldots = D_n = false, D_i = true\right) = q_i$$
    b. *We assume each cause is inhibited INDEPENDENTLY*. Thus the probability that we have D_i and D_j but not S is given by:
    $$P\left(S = false \mid D_1 = D_2 = \ldots = D_n = false, D_i = true, D_j = true\right) = q_i q_j$$
4. Thus, the entire conditional probability table can be fashioned with only n parameters $q_1, q_2, \ldots, q_n$ rather than $O(2^n)$.

Note: there is an alternative graphical model that captures these assumptions explicitly through auxiliary variables, but it's not important for our purpose.