# Dynamic Bayesian Networks

Luigi Portinale

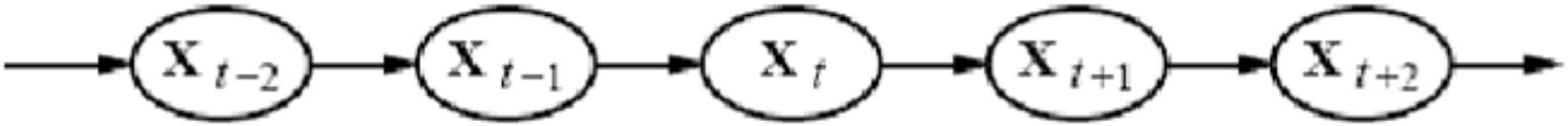
University of Piemonte Orientale, Italy

March 7-10, 2017

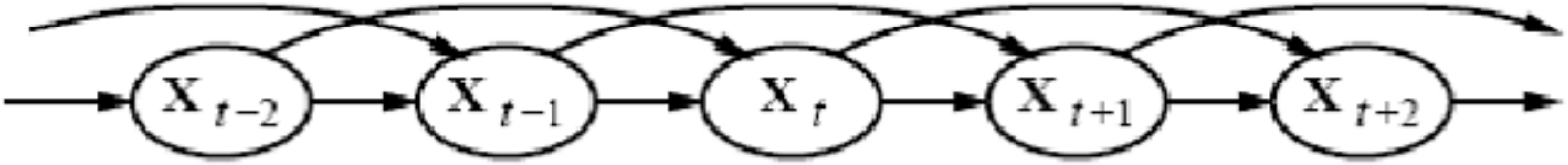
# Introduction

- Suppose to have a set of $N$ state variables: each state is the cross product of the variables
- Building a stochastic process $\rightarrow 2^N$ states (in case of binary variables)
- Dynamic Graphical Models allows for a factored representation of a stochastic process
- Let's first consider how to model a stochastic process via a (directed) PGM ...

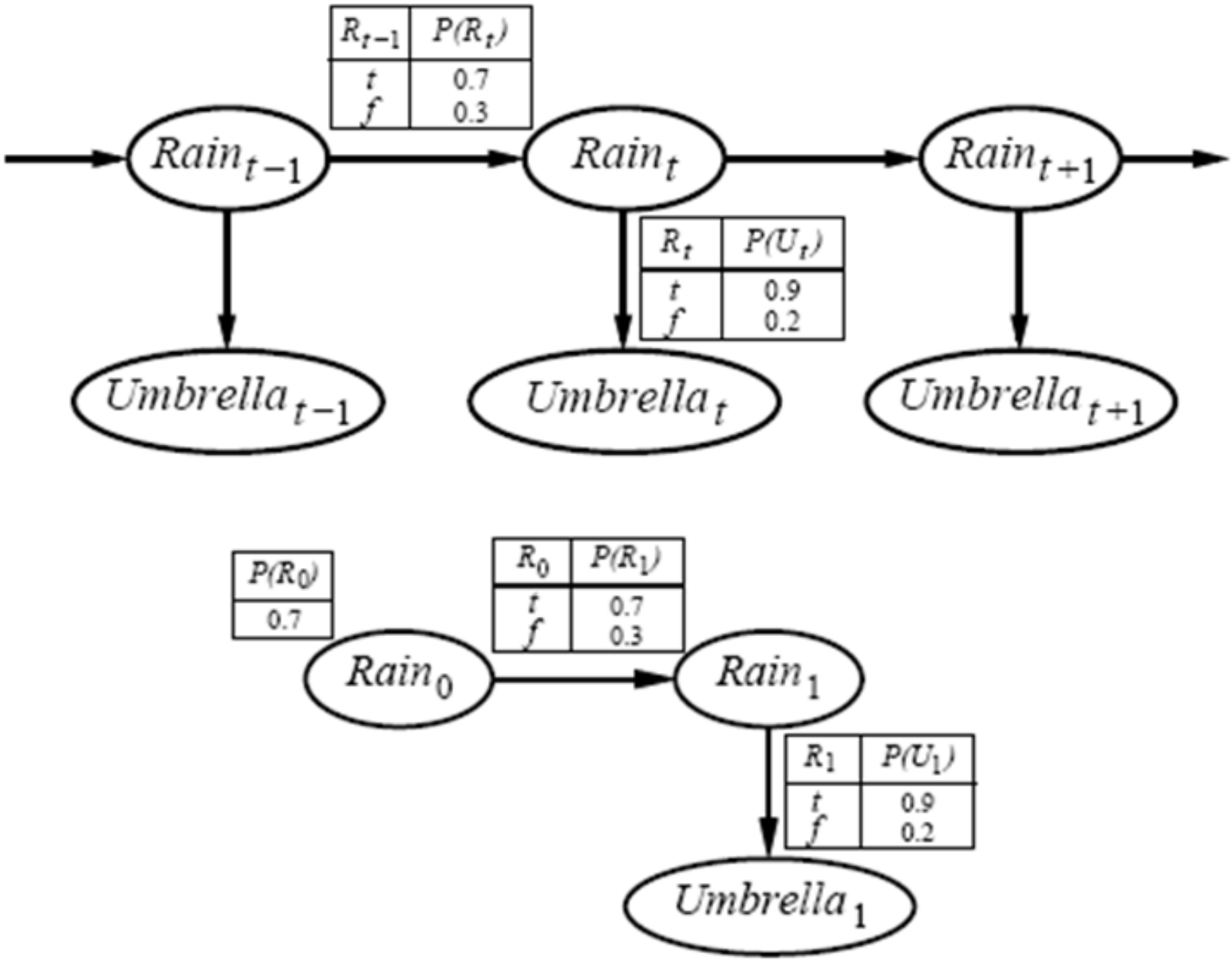
First-order Markov Process: $P(X_t|X_{0,t-1}) = P(X_t|X_{t-1})$




Second-Order Markov process:
$P(X_t|X_{0,t-1}) = P(X_t|X_{t-1}, X_{t-2})$




Stationary process: $P(X_t = j|X_{t-1} = j) = p_{ij}$ (transition independent from $t$)



# Dynamic Models

Let $\mathcal{X}^t$ be the set of variables at time $t$ and $\mathcal{X}_i^t$ a single variable at time $t$

## Trajectory

An assignment of values to each $\mathcal{X}_i^t$ for each relevant time $t$ is called a *trajectory*

## Discrete Time

We assume a time granularity $\Delta$, so we consider variables at discrete time points $t_0, t_0 + \Delta, \ldots t_0 + k\Delta \ldots$; we can then map time with $\mathbb{N} = 0, 1, \ldots k \ldots$

## Markovian Assumption

$(\mathcal{X}^{t+1} \perp \mathcal{X}^{(0:(t-1))} | \mathcal{X}^t)$

$\mathcal{P}(\mathcal{X}^0, \mathcal{X}^1, \ldots \mathcal{X}^T) = \mathcal{P}(\mathcal{X}^0) \prod_{t=0}^{T-1} \mathcal{P}(\mathcal{X}^{t+1} | \mathcal{X}^t)$

if stationary, transition model $\mathcal{P}(\mathcal{X}^{t+1} | \mathcal{X}^t) = \mathcal{P}(\mathcal{X}' | \mathcal{X})$

A little digression ...

## Conditional Bayesian Network

A *Conditional Bayesian Network* over $\mathcal{Y}$ given $\mathcal{X}$ is a BN with nodes $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$, where $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ disjoint. Variables in $X$ have no parents.

$$\mathcal{P}(\mathcal{Y}, \mathcal{Z}|\mathcal{X}) = \prod_{x \in \mathcal{Y} \cup \mathcal{Z}} P(x|\pi(x))$$

It follows that

$$\mathcal{P}(\mathcal{Y}|\mathcal{X}) = \sum_{\mathcal{Z}} \mathcal{P}(\mathcal{Y}, \mathcal{Z}|\mathcal{X})$$

## Back to temporal models !

### 2-TBN

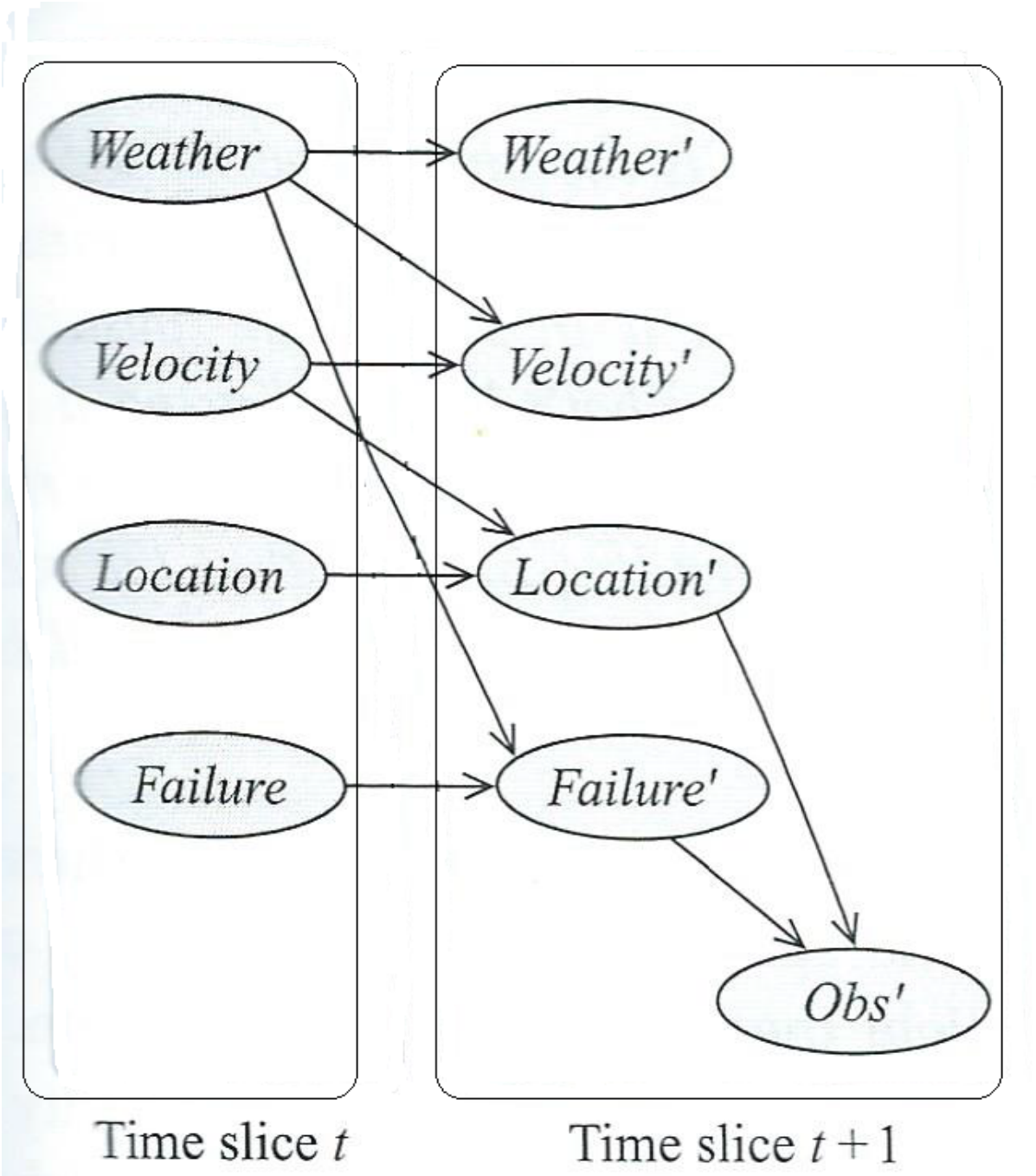
A *2-time-slice Bayesian Network* (2-TBN) for a process over $\mathcal{X}$ is a conditional BN over $\mathcal{X}'$ given $\mathcal{X}_I$, where $\mathcal{X}_I \subseteq \mathcal{X}'$ are called *interface variables*

Variables in $\mathcal{X}'$ have parents, while interface variables are those variables whose values at time $t$ have a direct effect on variables at time $t+1$.

Time $t$ is called the *anterior layer* and $t+1$ the *ulterior layer*

Interface variables are in the anterior and $\mathcal{X}'$ are in the ulterior layer

Only interface variables can be parents of variables in $\mathcal{X}'$

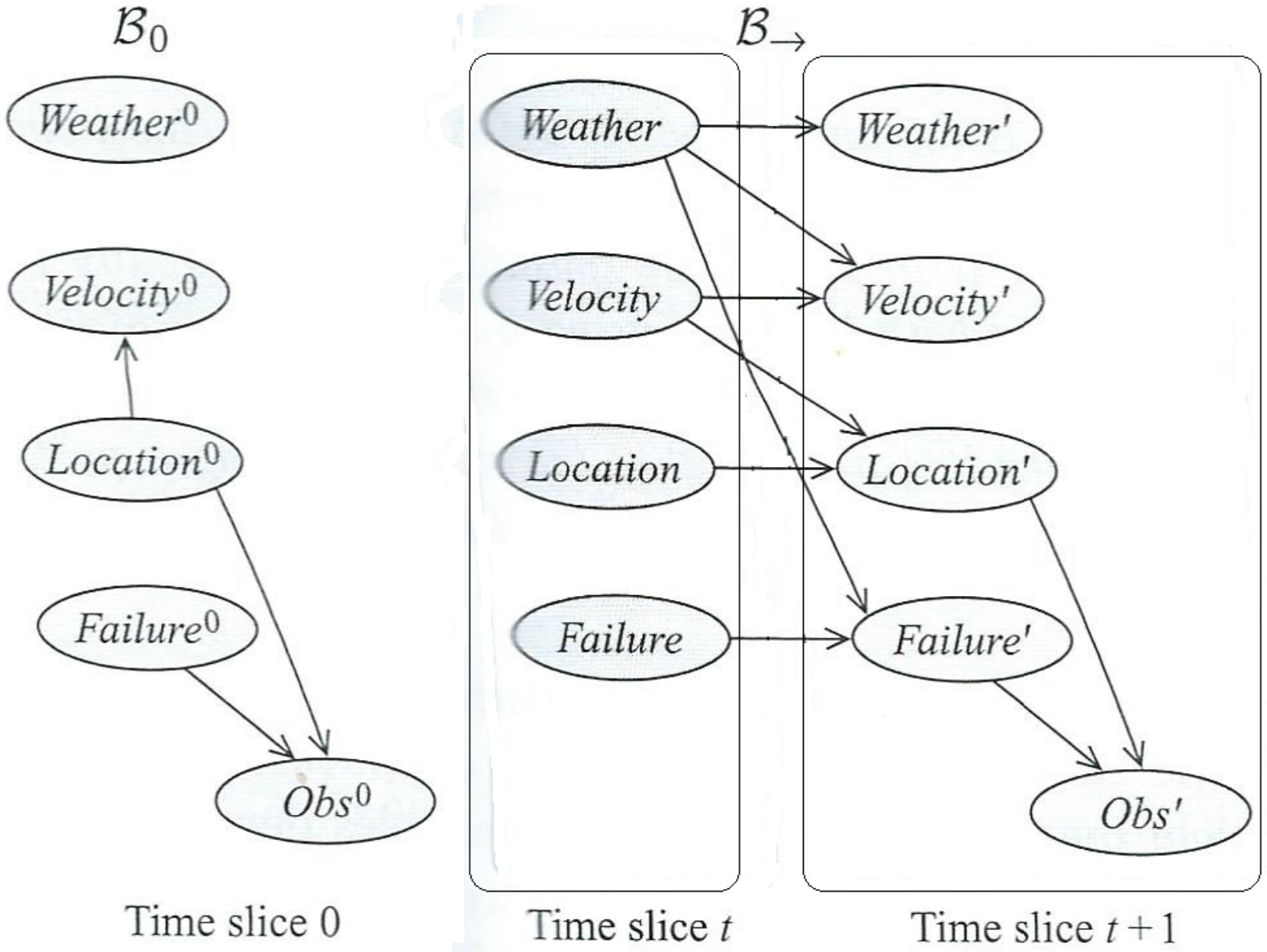


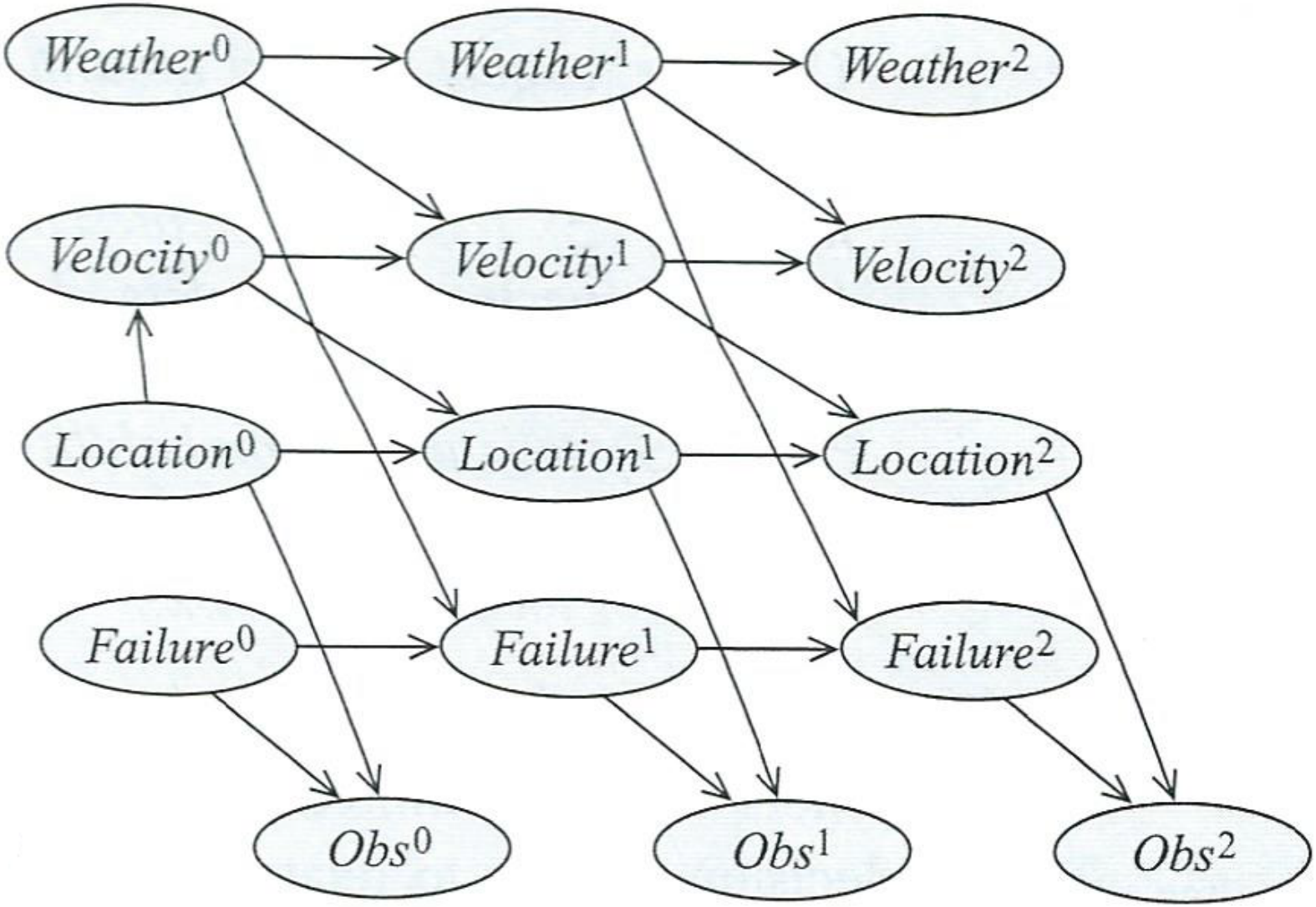
- each variable $\mathcal{X}_i$ is actually a *template variable*, representing different instances of $\mathcal{X}_i$ at different time points
- the CPD $\mathcal{P}(\mathcal{X}'_i|\pi(\mathcal{X}'_i))$ is a *template factor*
- $\mathcal{X}_i$ and $\mathcal{X}'_i$ represent the same variable at the anterior and ulterior layer respectively
- arcs whithin the same layer are called *intra-slice*, arcs between layers are called *inter-slice*
- in a *canonical representation* there are no intra-slice arcs in the anterior layer
- inter-slice arcs between $\mathcal{X}$ and $\mathcal{X}'$ are called *persitence arcs* and $\mathcal{X}$ is a *persistent variable*
- because of stationarity, given the initial distribution we can *unroll* (i.e. replicate) the 2-TBN over sequences of any length

## Dynamic Bayesian Network

A *Dynamic Bayesian Network* (DBN) is a pair $\langle \mathfrak{B}_0, \mathfrak{B}_{\rightarrow} \rangle$ where $\mathfrak{B}_0$ is a BN over $\mathcal{X}^0$ representing the initial distribution and $\mathfrak{B}_{\rightarrow}$ is a 2-TBN. For any time span $T \geq 0$, the distribution $\mathcal{X}^{(0:T)}$ is defined as an unrolled BN where, for any $i = 1 \ldots n$

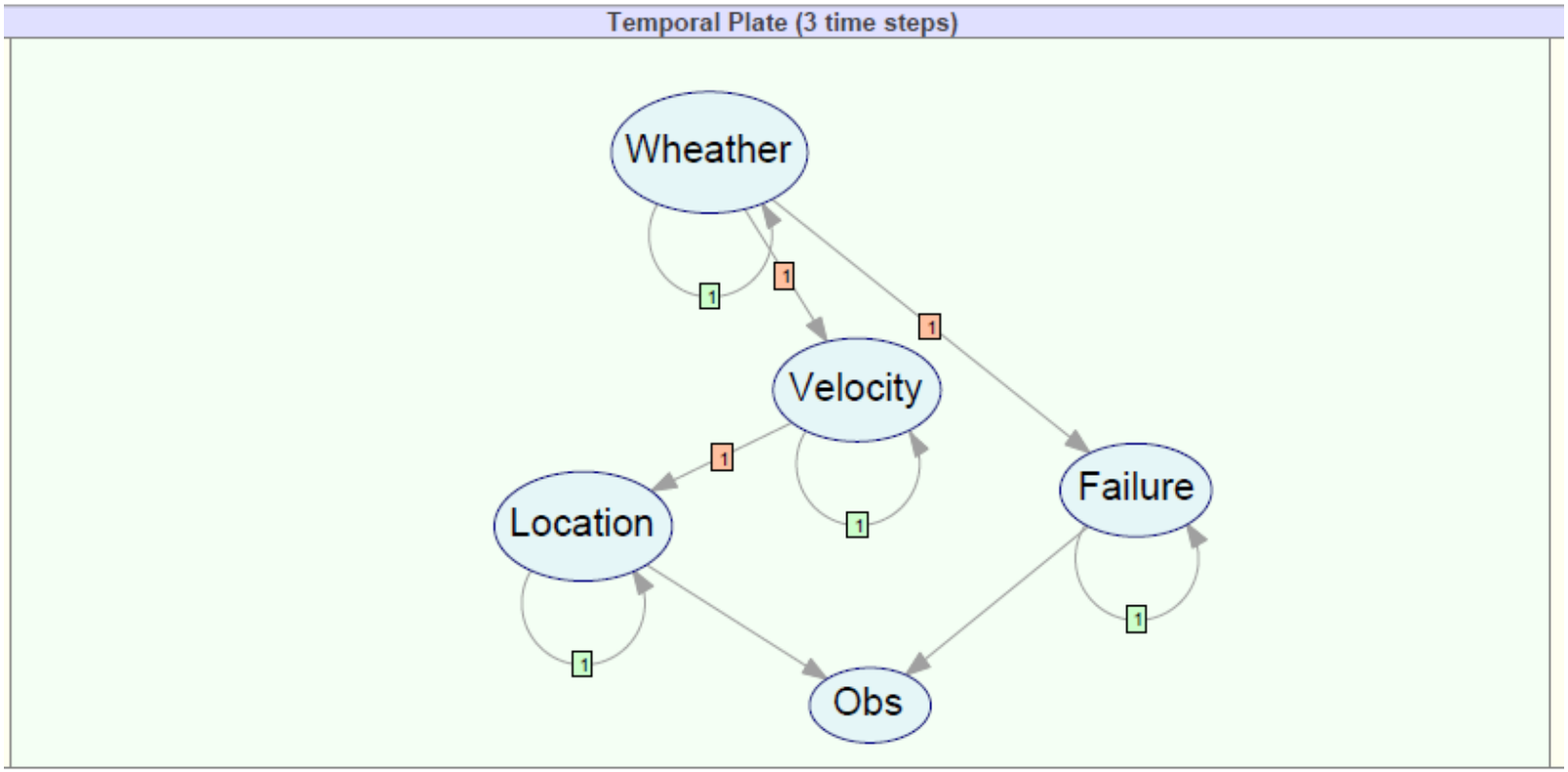
- structure and CPDs of $\mathcal{X}_i^0$ are those for $\mathcal{X}_i$ in $\mathfrak{B}_0$
- strcture and CPDs of $\mathcal{X}_i^t$ $(t > 0)$ are those for $\mathcal{X}_i'$ in $\mathfrak{B}_{\rightarrow}$



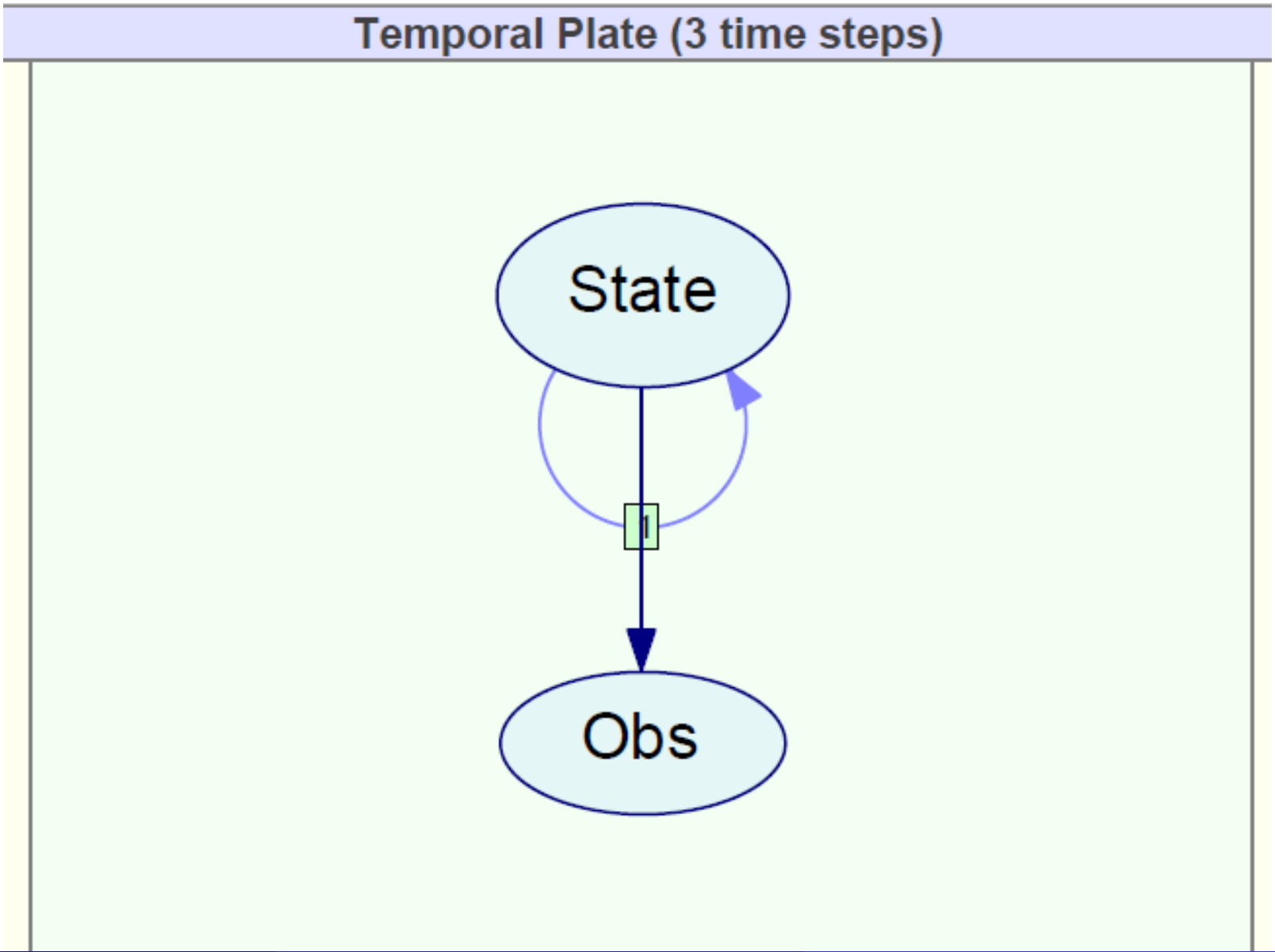



DBN unrolled over 3 steps

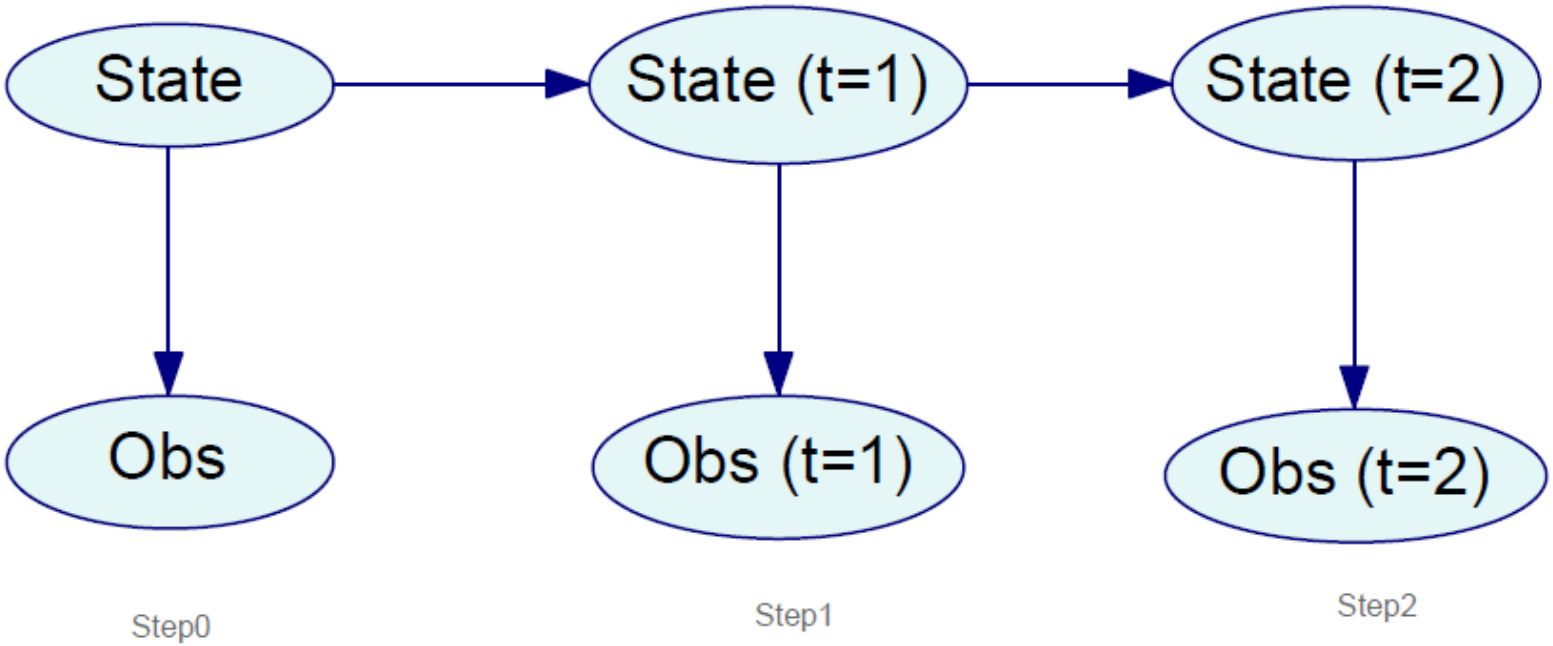
# Alternative Representation 2-TBN: Plate Model



# Hidden Markov Model (HMM)



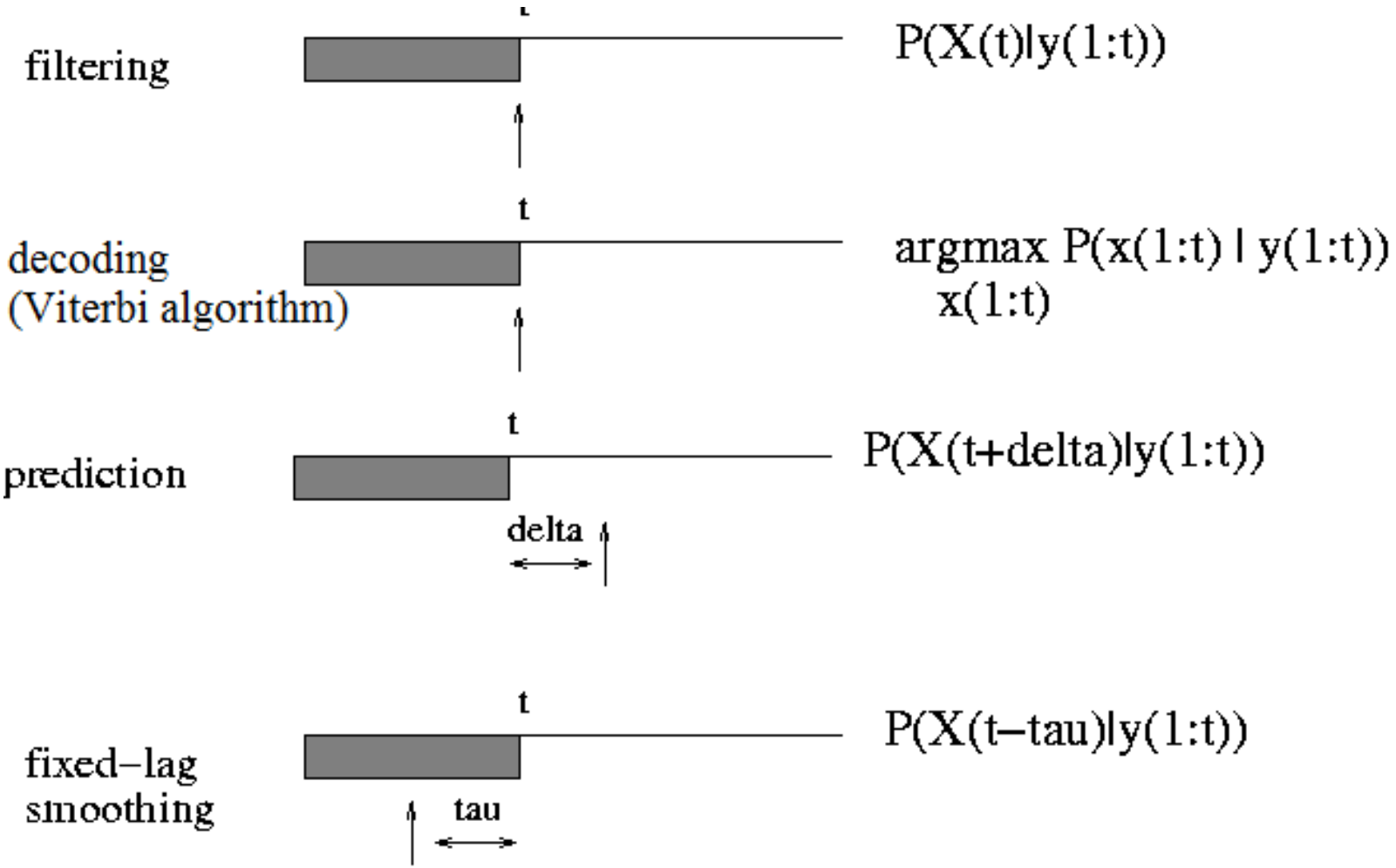
# HMM: Unrolled version



# Inference in DBN

## Inference Tasks

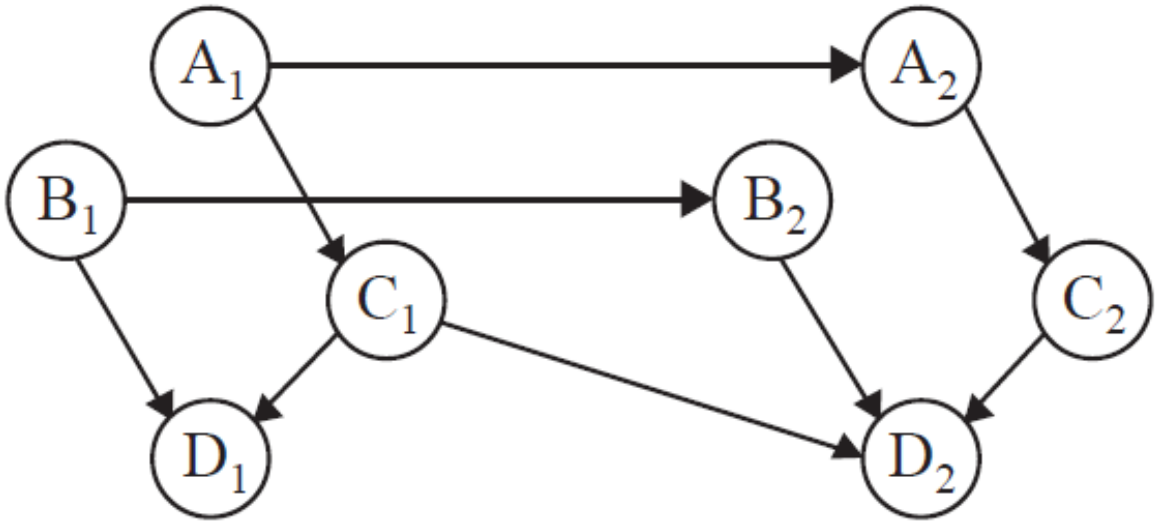
- **Filtering** or **Monitoring**: computing $P(X^t|y_{1:t})$, i.e. tracking the probability of the system state taking into account the stream of observations received.
- **Prediction**: computing $P(X^{t+h}|y_{1:t})$ for some horizon $h > 0$, i.e. predicting a future state taking into consideration the observation up to now (filtering is a special case of prediction with $h = 0$).
- **Smoothing**: computing $P(X^{t-l}|y_{1:t})$ for some $l < t$, i.e. estimating what happened $l$ steps in the past given all the evidence (observations) up to now.
- **Decoding**: computing $\arg\max_{x_{1:t}} P(x_{1:t}|y_{1:t})$ i.e., determining the most probable sequence of states, given the stream of observations



# Exact Inference

- it is possible to extend the Belief Propagation algorithm on Junction Tree to DBN: *1.5 JT algorithm* [Murphy:2002]
- the interface variables $\mathcal{X}_I$ (at the anterior layer $t$) d-separate variables at time $t' < t$ and non interface variables at time $t$ from variables at time $t'' > t$

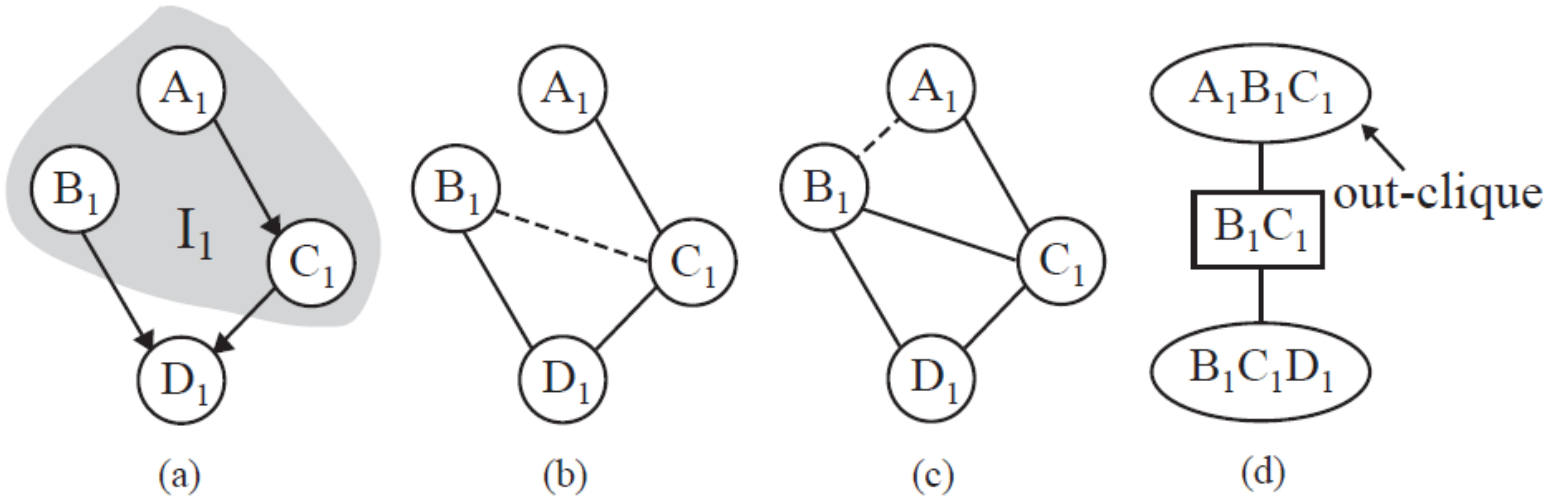
Thus $\mathcal{X}_I$ is a sufficient statistics for all the trajectories up to time $t$



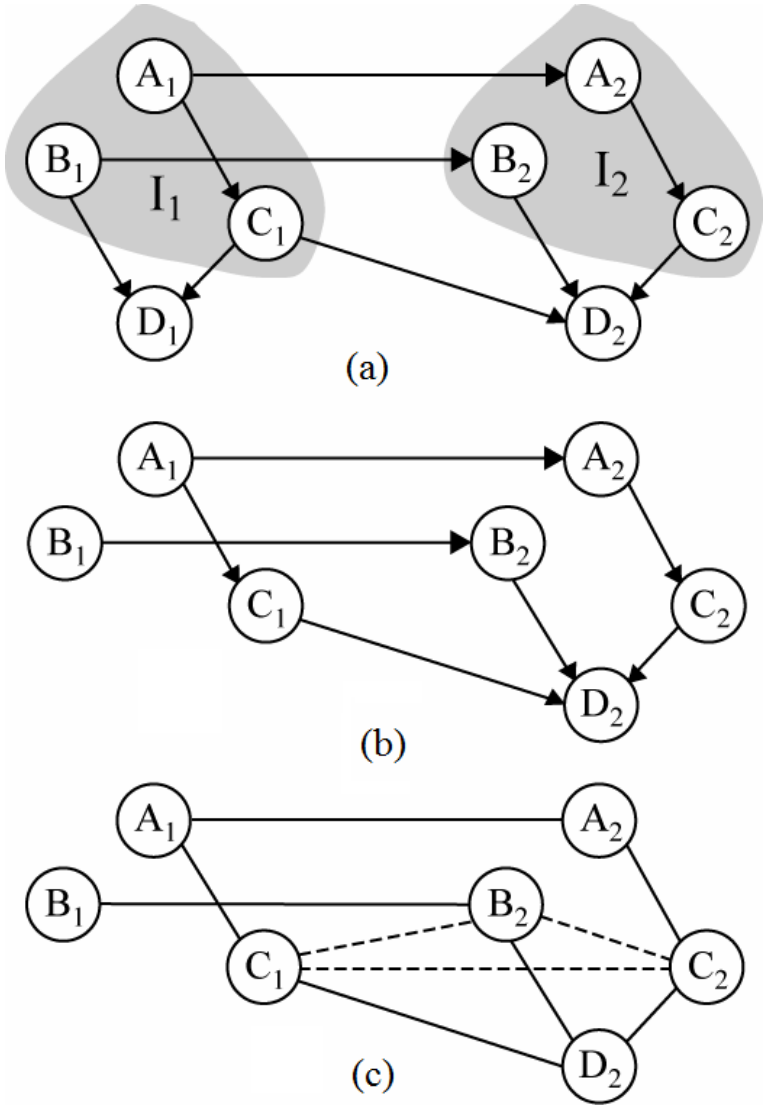

$\mathcal{X}_I = \{A_1, B_1, C_1\}$; once we know such nodes at time $t$, all other nodes at $t$ ($D_1$) and all nodes at $t' < t$ are irrelevant for future states at $t'' > t$

- construction of two different JT: $J_0$ for the initial instant; $J_t$ for subsequent time instants
- change wrt static JT inference: *connect all the interface nodes if they are not*
- previous step assures that interface nodes are in the same clique; since they are the sufficient statistics, the interface clique is used to connect different time points
- in the following subscript $_1$ refer to anterior and subscript $_2$ to ulterior layers

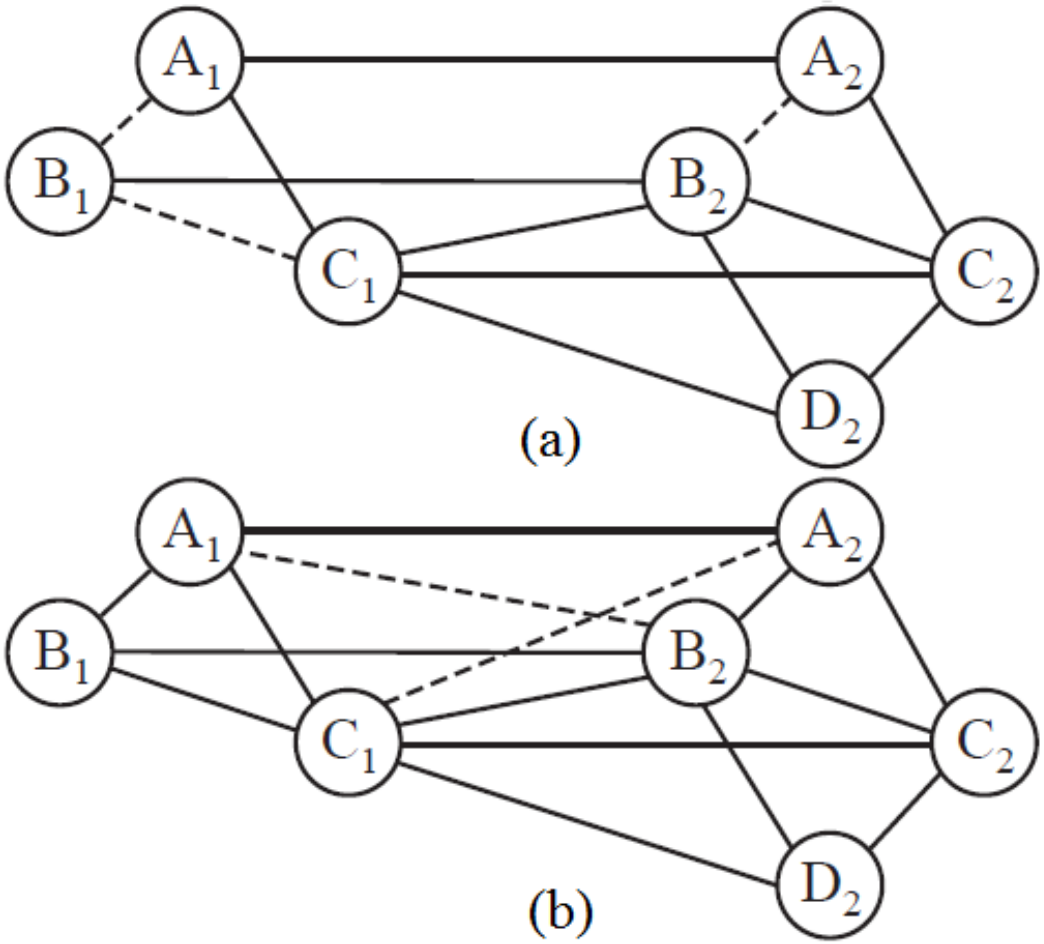
# Construction of $J_0$



# Construction of $J_t$: preliminary steps



# Construction of $J_t$: final steps




Cliques:

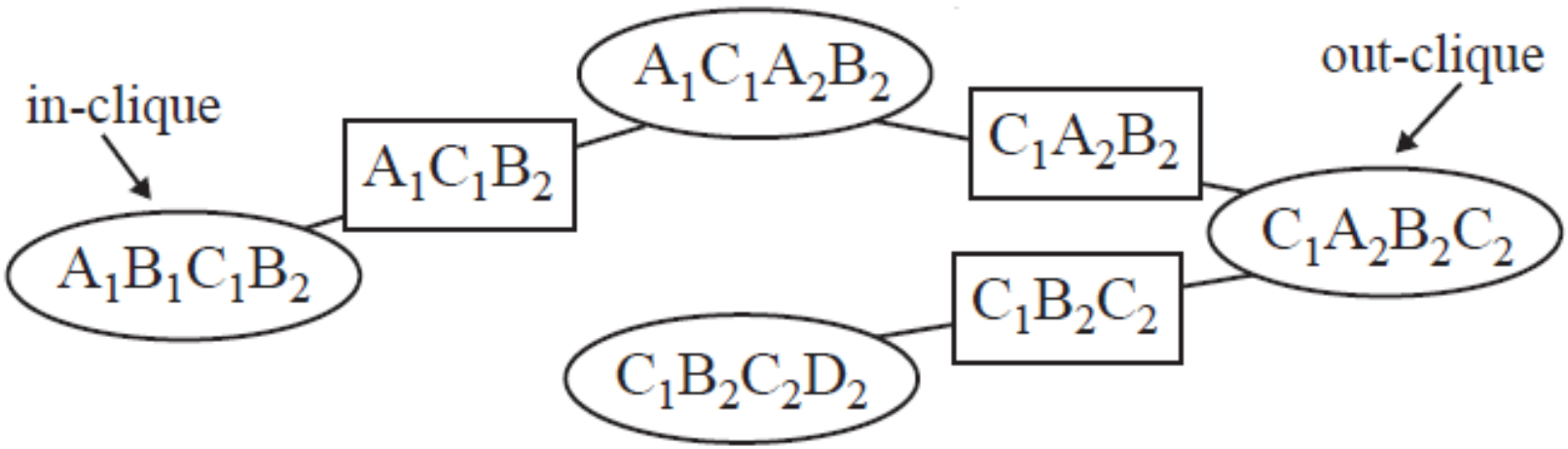
$A_1B_1C_1B_2$

$A_1C_1A_2B_2$

$C_1A_2B_2C_2$

$C_1B_2C_2D_2$

# Construction of $J_t$: the resulting junction tree

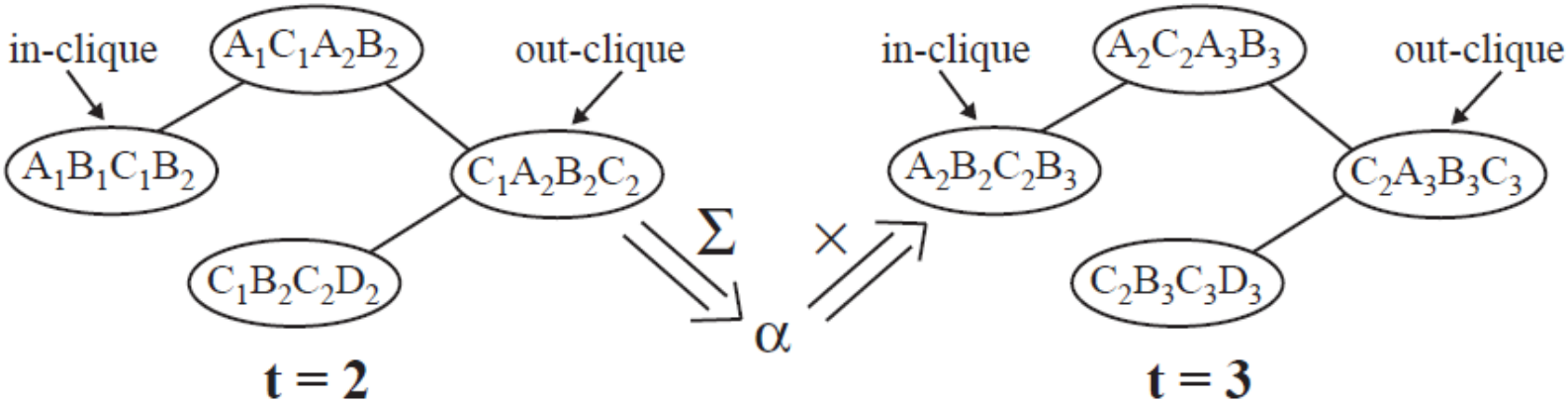


# The 1.5JT Algorithm

- **Initialization.** Create $J_0$ and $J_t$;
- **Evidence incorporation.** If evidence is at time $t = 0$ then incorporate it into $J_0$; if evidence is at time $t > 0$, then incorporate it into $J_t$ at time $t$;
- **Query at time** $t = 0$**.** Execute belief propagation on $J_0$ and marginalize on the queried variables;
- **Query at time** $t > 0$**.** Execute belief propagation on $J_t$ and marginalize on the queried variables. Consider now the in-clique $C_1$ and the out-clique $C_2$ of $J_t$; compute factor $\alpha_t = \sum_{C_2 \setminus I_2} \Phi(C_2)$, increment time to $t + 1$, reset potentials and update $\Phi_{C_1} := \alpha_t \Phi_{C_1}$.

This implements filtering/prediction (slight variations for smoothing)

# Inference advancement




1. $\alpha = \phi_{A_2B_2C_2} = \sum_{C_1} \phi_{C_1A_2B_2C_2}$
2. time incremented, potentials reset
3. $\phi_{A_2B_2C_2B_3} = \alpha * \phi_{A_2B_2C_2B_3}$

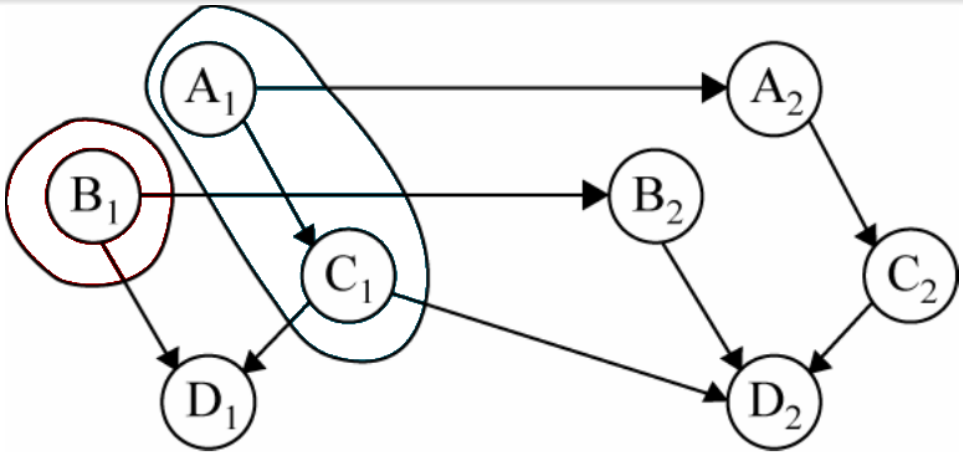
# Approximate Inference

- Approximante inference can be dealt with in two different ways also for DBN
- Sampling approaches: estimate the desired posterior through sampling from trajectories. Problem to address: how to deal with arbitrarly large (even unbounded) trajectories
- Variational approaches: approximate the correct distribution with a simpler one, easier to manage and compute.

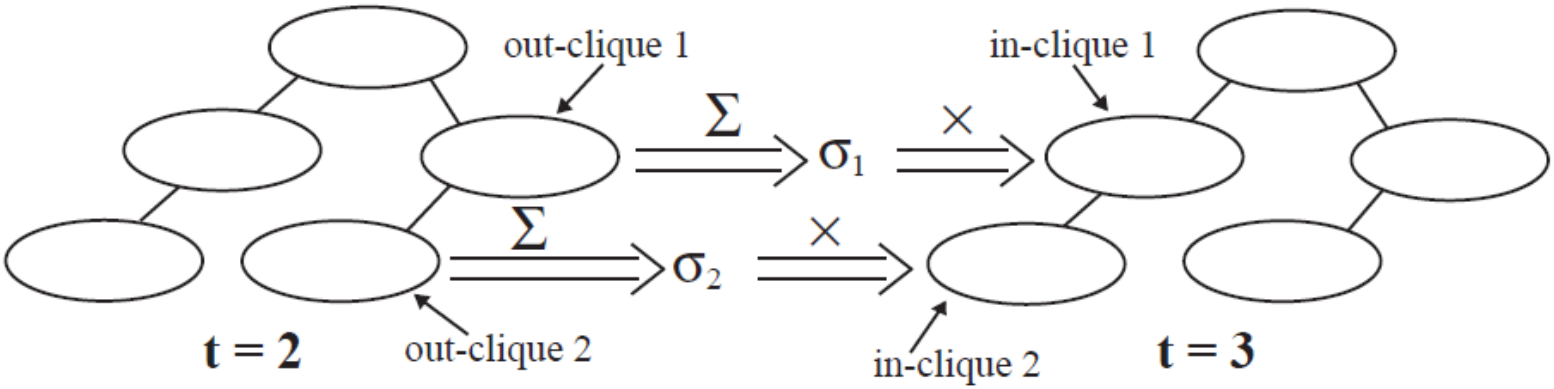
We discuss here a variational algorithm, which is based on JT propagation, and having 1.5JT as a special case [Boyen & Koller: 1998]

## Boyen-Koller Algorithm (BK)

- Main idea: since the set of interface variables may be very large, partition it into subsets of interface variables called *clusters*.
- it follows that also the interface clique can be divided into smaller subcliques
- approximate the *right* interface clique potential with the product of the potentials of the subcliques corresponding to clusters




$$\phi_{A_1B_1C_1} \approx \phi_{A_1C_1}\phi_{B_1}$$




1. $\sigma_1 = \sum \phi_{\text{out-clique 1}}$
   $\sigma_2 = \sum \phi_{\text{out-clique 2}}$
2. time incremented, potentials reset
3. $\phi_{\text{in-clique 1}} = \sigma_1 * \phi_{\text{in-clique 1}}$
   $\phi_{\text{in-clique 2}} = \sigma_2 * \phi_{\text{in-clique 2}}$

# BK Algorithm

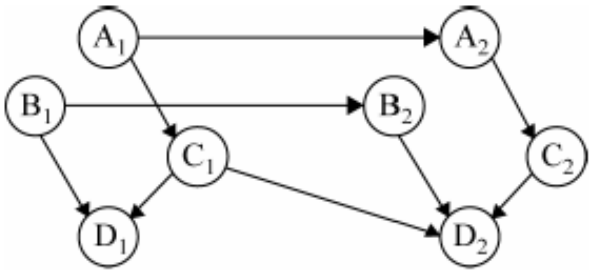
- 1.5JT is a special case of BK when there is only one single cluster containing the whole interface
- the most aggressive approximation is the *fully factorized BK*, with as many clusters as many interface variables, each cluster being a singleton
- accuracy can be measured by means of *Kullback-Leibler Divergence*, measuring how many information is lost in using distribution $\mathcal{Q}$ in place of $\mathcal{P}$
- best accuracy is obtained if no node in any one cluster is a parent of a node in a different cluster whithin a single time slice

$$\mathcal{D}_{KL}(P\|Q) = \sum_i \mathcal{P}(i) \log \frac{\mathcal{P}(i)}{\mathcal{Q}(\rangle)}$$

# BK Algorithm Example Results

| Average KL-Divergence for Various BK Cluster Factorizations | | | | | |
|---|---|---|---|---|---|
| | ABC | AC \| B | AB \| C | A \| BC | A \| B \| C |
| KL-Divergence | 0 | 6.796e-08 | 3.482e-04 | 3.518e-04 | 3.489e-04 |
| Time (s) | 8.2291 | 7.6949 | 7.9081 | 7.9741 | 7.3594 |

- Run for 10,000 timesteps, random evidence applied to node D on each timestep
- Nodes queried: A, B, C
- KL-Divergence shown is absolute value of KL-D from truth of queried distributions, averaged over A, B, and C




- Time given is for execution of all 10,000 timesteps and is averaged over 15 runs
- As shown, best speed obtained with fully-factored approximation and best accuracy obtained when no node in any one cluster is parent of a node in a different cluster within a single timeslice (ABC, AC | B)
- The interface algorithm is simply a special case of BK with no independence assumed (ABC)