Thin Junction Tree Filters for Simultaneous Localization and Mapping

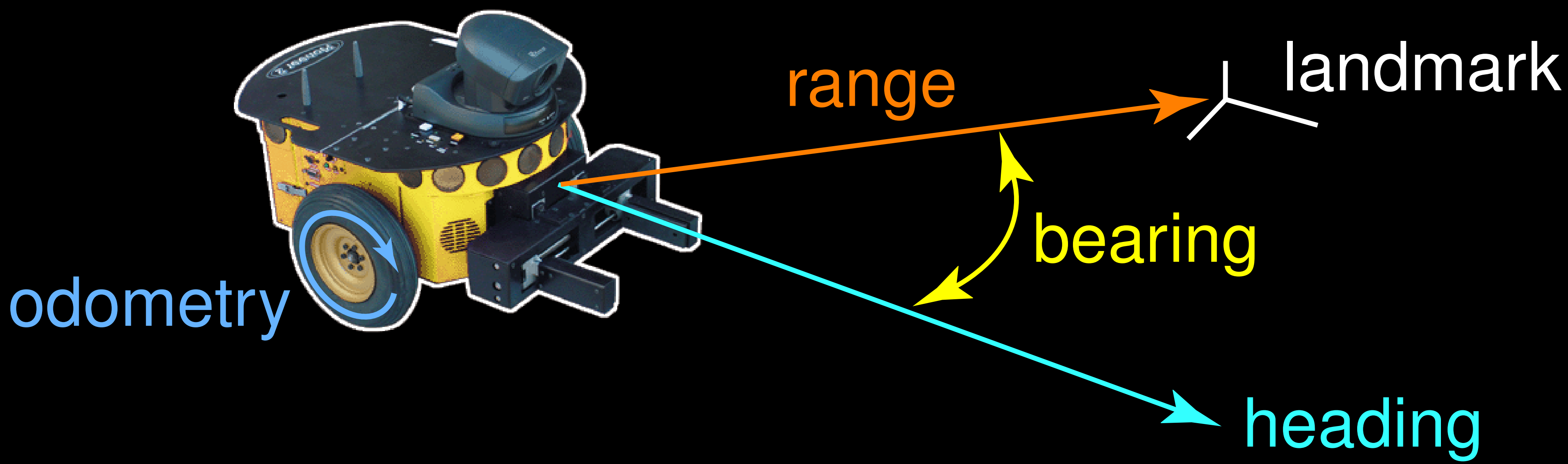
Mark Paskin

Computer Science Division
University of California, Berkeley
mark@paskin.org

Simultaneous Localization and Mapping

A mobile robot navigating in an unknown environment must incrementally

1. build a map of its surroundings and
2. localize itself within that map.



The traditional approach: Kalman filters

- View SLAM as a state estimation problem in a linear-Gaussian dynamical system

The traditional approach: Kalman filters

- View SLAM as a state estimation problem in a linear-Gaussian dynamical system
- System state at time t: $[R_t; L_1; \ldots; L_{N_t}]$

The traditional approach: Kalman filters

- View SLAM as a state estimation problem in a linear-Gaussian dynamical system
- System state at time t: $[R_t; L_1; \ldots; L_{N_t}]$
- The belief state is a Gaussian $\mathcal{N}(\mu_t, \Sigma_t)$

The traditional approach: Kalman filters

- View SLAM as a state estimation problem in a linear-Gaussian dynamical system
- System state at time t: $[R_t; L_1; \ldots; L_{N_t}]$
- The belief state is a Gaussian $\mathcal{N}(\mu_t, \Sigma_t)$
- Time & space complexity: $\Theta(N_t^2)$

Thin junction tree filters

- `TJTF`: a novel algorithm for approximate filtering in dynamic Bayesian networks

Thin junction tree filters

- TJTF: a novel algorithm for approximate filtering in dynamic Bayesian networks
- When applied to the SLAM problem, we obtain space complexity: $O(N_t)$

Thin junction tree filters

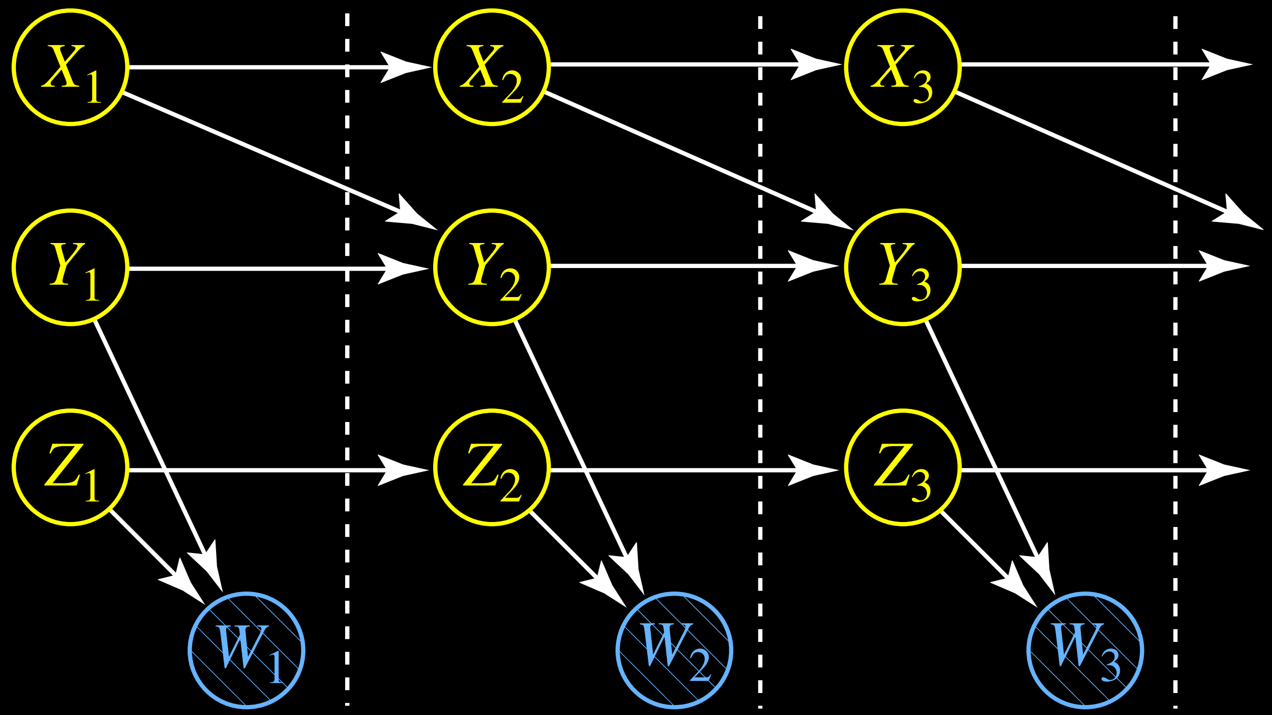
- TJTF: a novel algorithm for approximate filtering in dynamic Bayesian networks
- When applied to the SLAM problem, we obtain

  space complexity: $O(N_t)$

  time complexity: $O(N_t)$ or $O(1)$

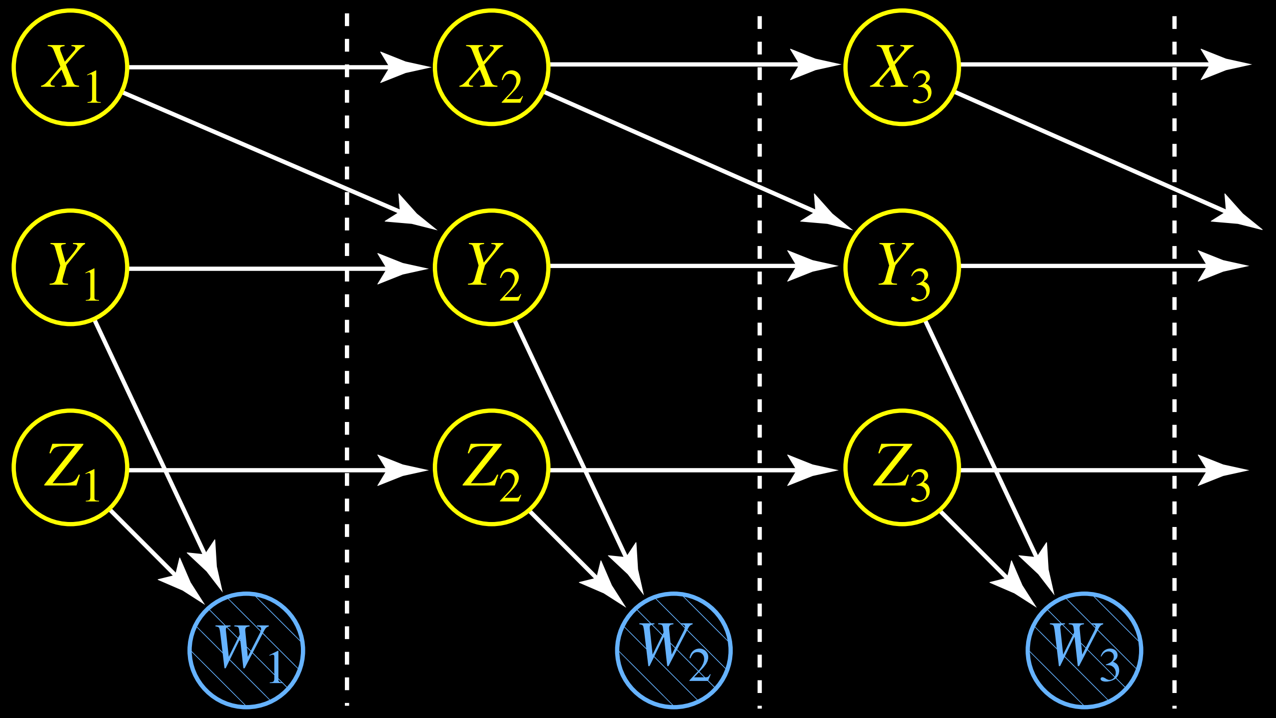
Filtering in dynamic Bayesian networks

A dynamic Bayesian network (DBN) is a compact representation of a complex stochastic process.



Filtering in dynamic Bayesian networks

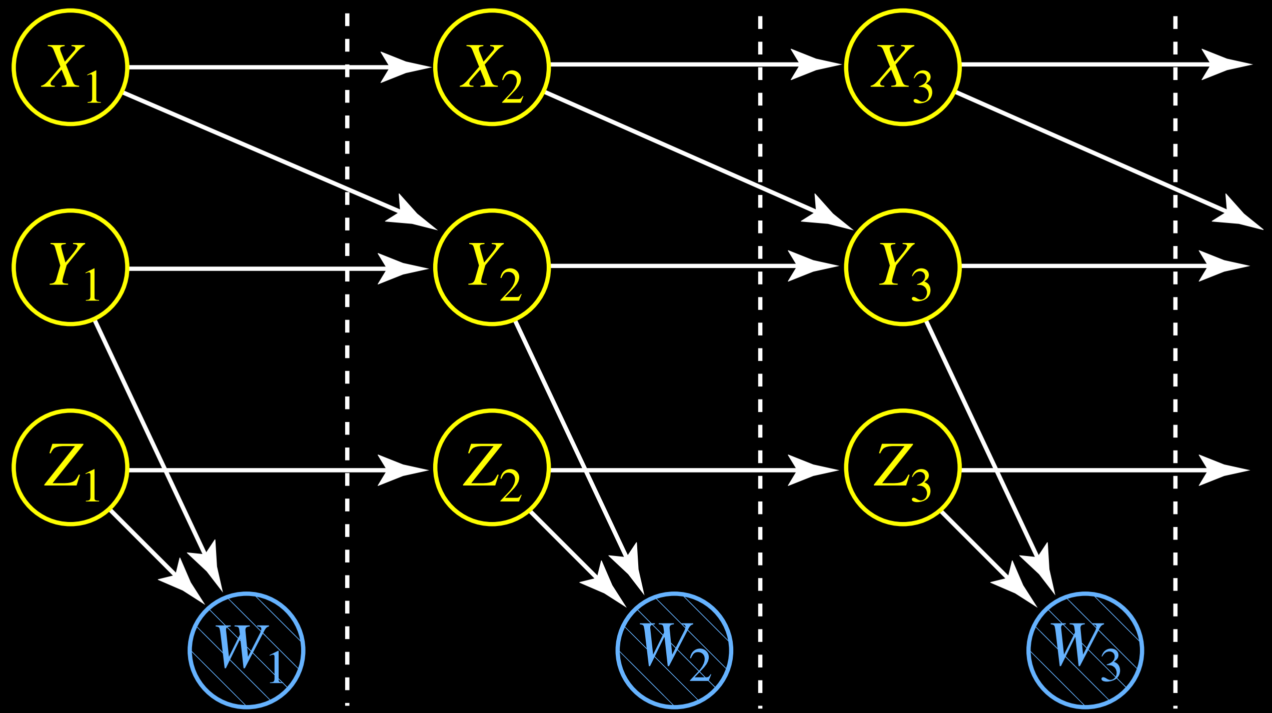
A dynamic Bayesian network (DBN) is a compact representation of a complex stochastic process.



Belief state at time t: $b_t = p(x_t, y_t, z_t \mid \overline{w}_{1:t-1})$

Filtering in dynamic Bayesian networks

A dynamic Bayesian network (DBN) is a compact representation of a complex stochastic process.



Belief state at time t: $b_t = p(x_t, y_t, z_t \mid \overline{w}_{1:t-1})$

Filtering: iteratively update $b_t \xrightarrow{\overline{w}_t} b_{t+1}$

Complexity of filtering in DBNs

The DBN is compact, but the belief state is not:

$$p(x_1)p(y_1)p(z_1)$$

Complexity of filtering in DBNs

The DBN is compact, but the belief state is not:

$$p(x_1)p(y_1)p(z_1) \qquad p(x_2, y_2, z_2 \mid \overline{w}_1)$$

Complexity of filtering in DBNs

The DBN is compact, but the belief state is not:

$p(x_1)p(y_1)p(z_1)$ $\qquad$ $p(x_2, y_2, z_2 \mid \overline{w}_1)$

Exact filtering in DBNs is intractable.

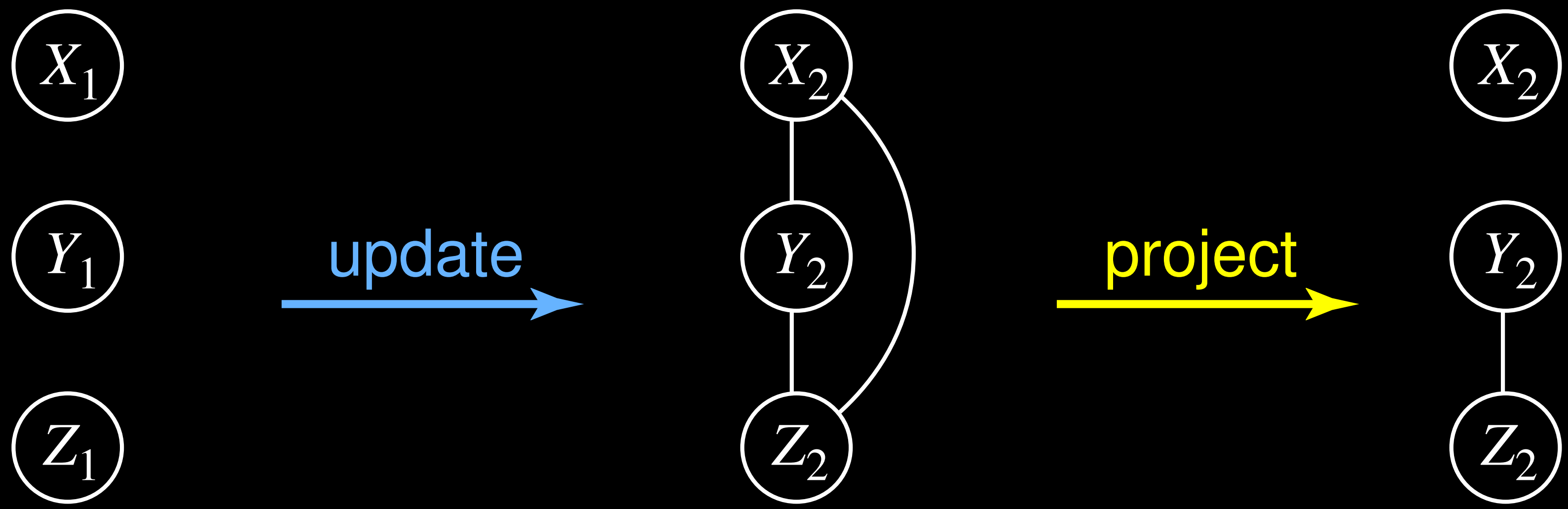
The Boyen & Koller (1998) Algorithm

Choose a fixed, tractable form for the belief state and project to the closest density of that form:

$p(x_1)p(y_1)p(z_1) \quad p(x_2, y_2, z_2 \mid \overline{w}_1)$

The Boyen & Koller (1998) Algorithm

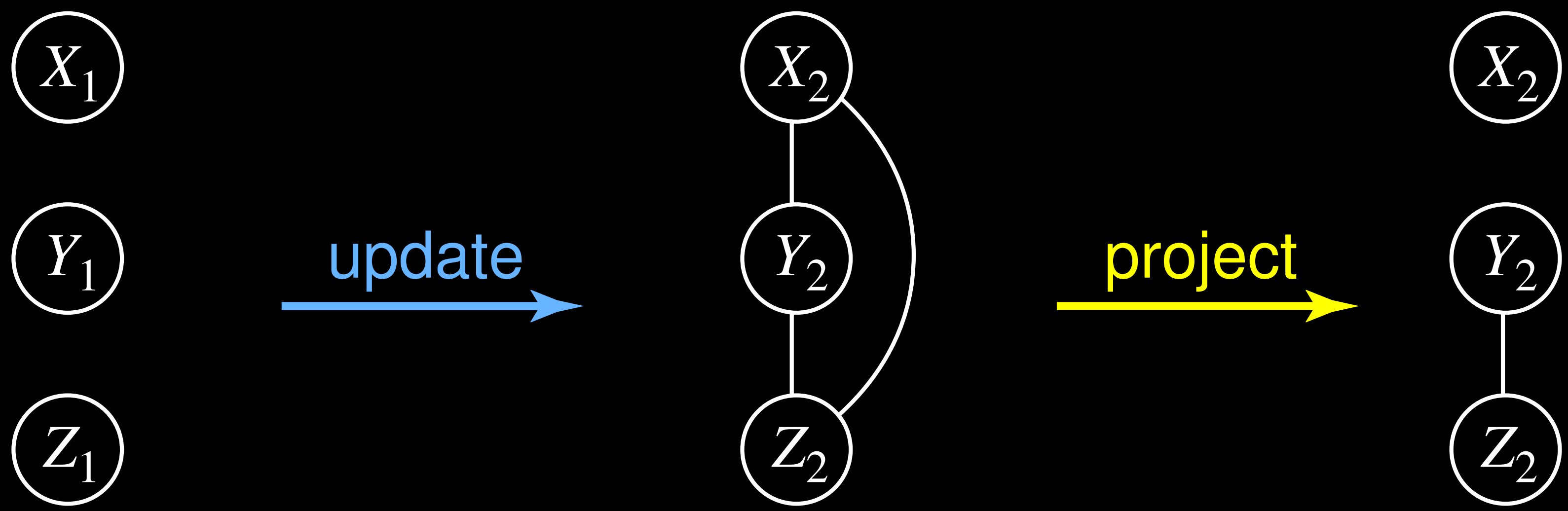
Choose a fixed, tractable form for the belief state and project to the closest density of that form:



$p(x_1)p(y_1)p(z_1)$ $\quad$ $p(x_2, y_2, z_2 \mid \overline{w}_1)$ $\quad$ $p(x_2 \mid \overline{w}_1) \times p(y_2, z_2 \mid \overline{w}_1)$

The Boyen & Koller (1998) Algorithm

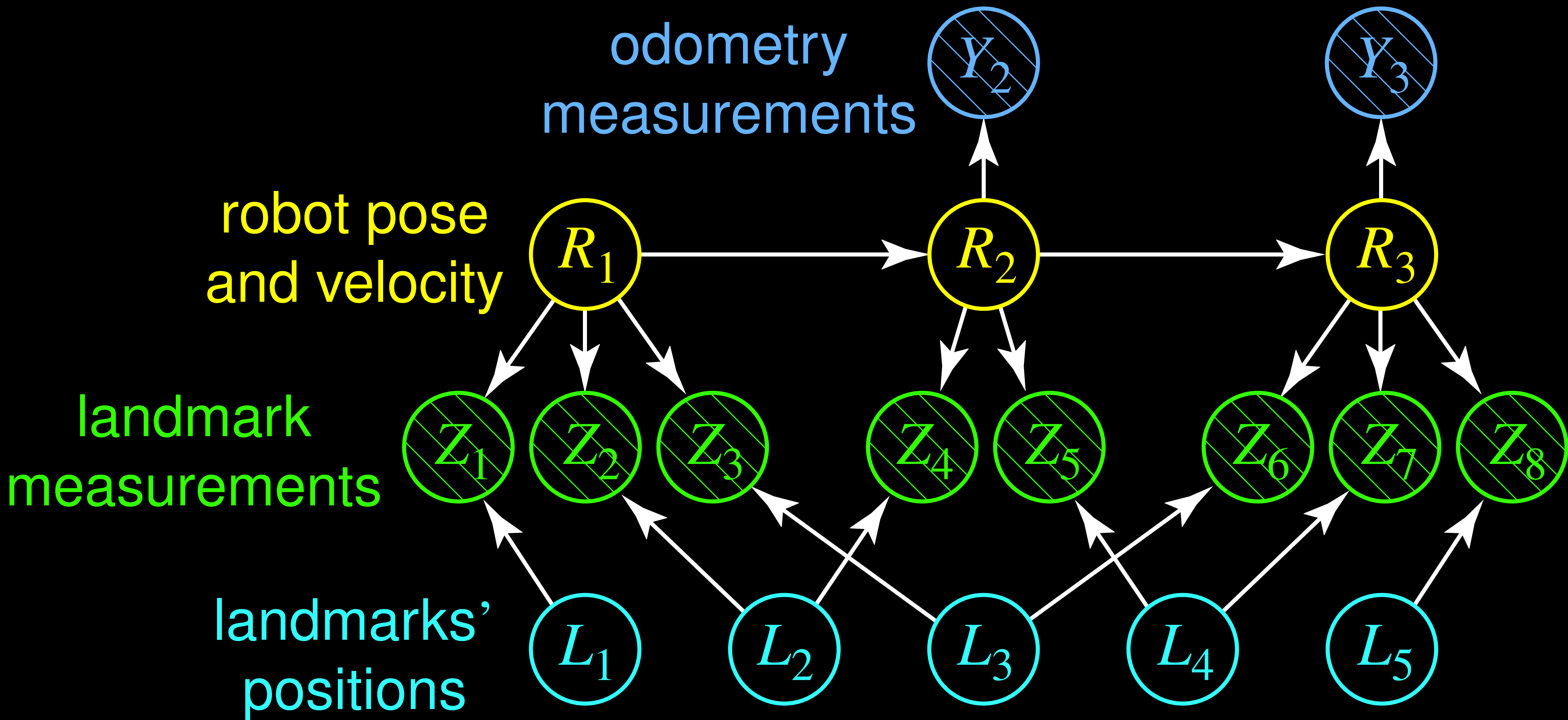
Choose a fixed, tractable form for the belief state and project to the closest density of that form:



$p(x_1)p(y_1)p(z_1)$ $\quad$ $p(x_2, y_2, z_2 \mid \overline{w}_1)$ $\quad$ $p(x_2 \mid \overline{w}_1) \times p(y_2, z_2 \mid \overline{w}_1)$

Problem: what is the best tractable form?

An example SLAM DBN



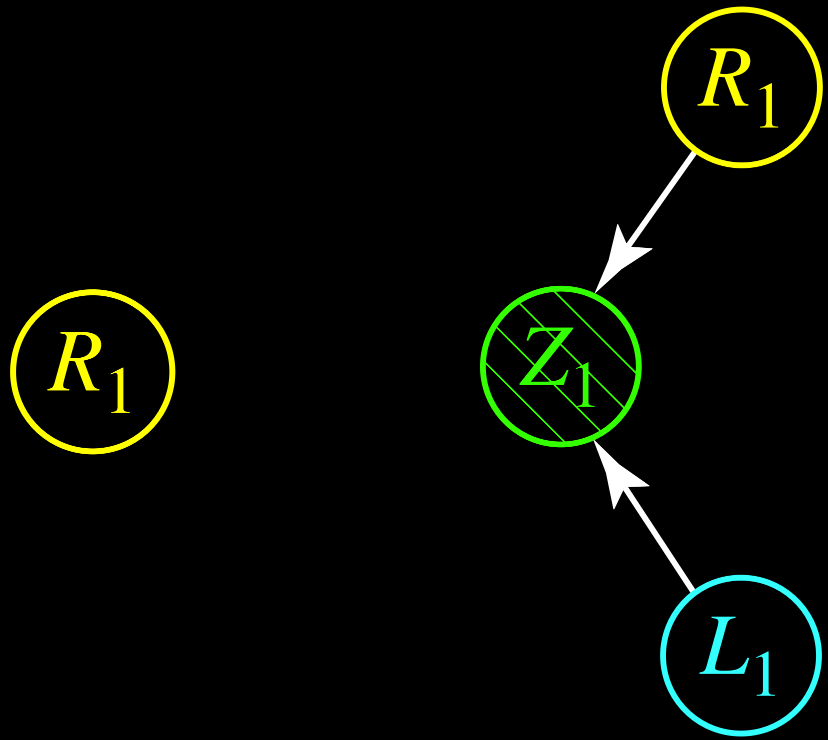
Filtering the SLAM DBN: estimation

$$p(r_1)$$

R_1

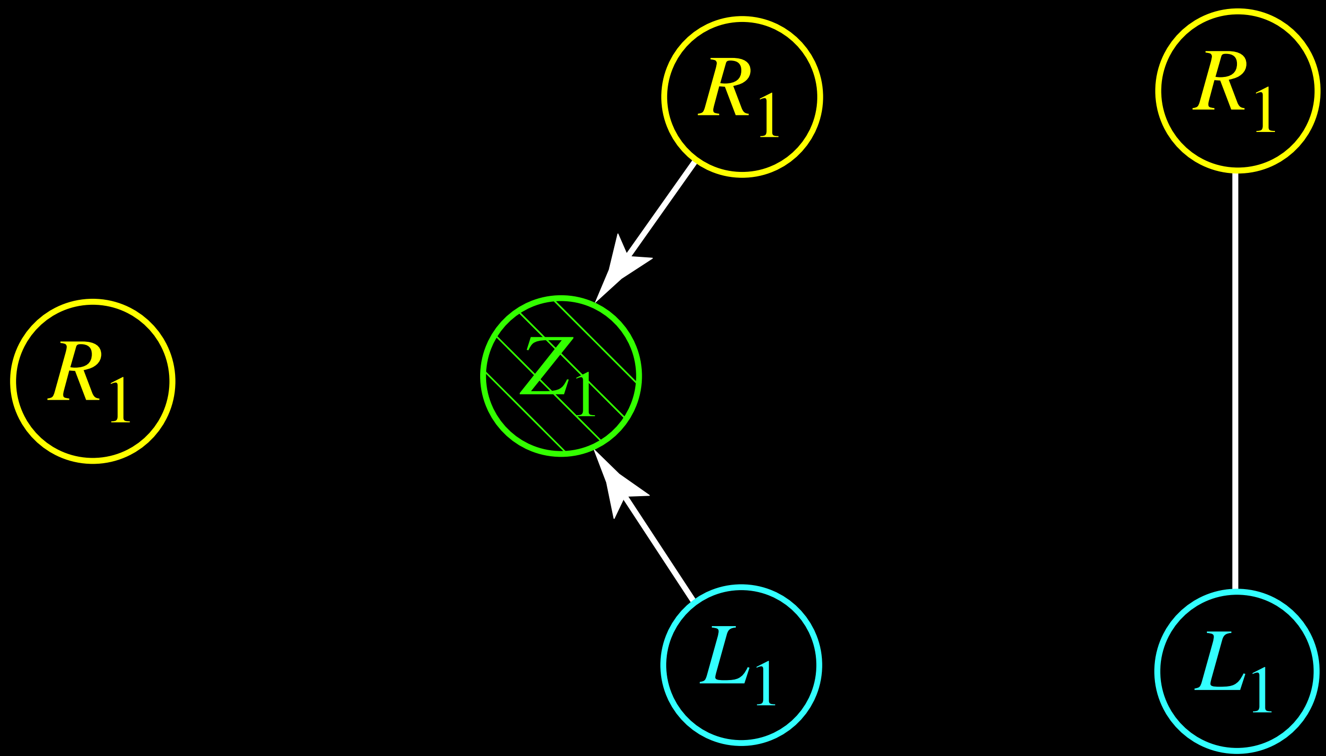
Filtering the SLAM DBN: estimation

$$p(r_1, l_1 \mid \overline{z}_1) \propto p(r_1) \cdot p(l_1) \cdot p(\overline{z}_1 \mid r_1, l_1)$$



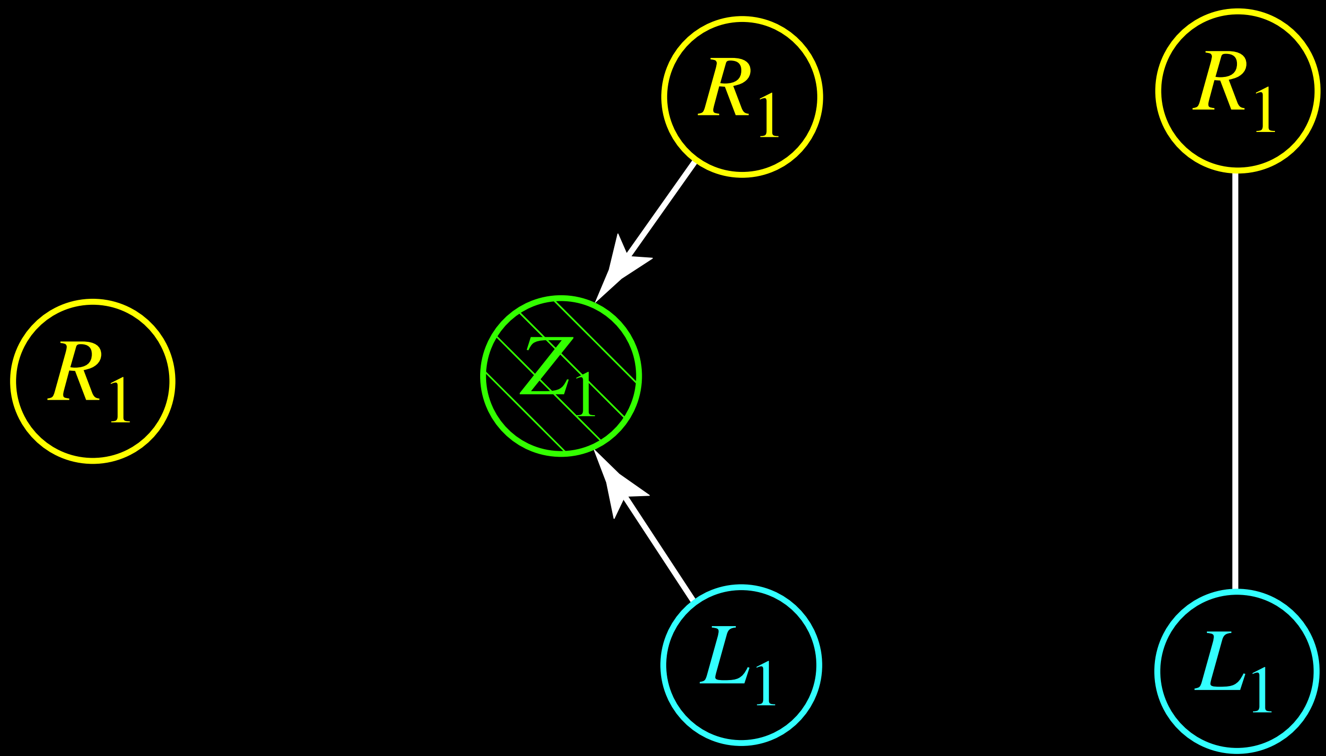
Filtering the SLAM DBN: estimation

$$p(r_1, l_1 \mid \overline{z}_1) \propto p(r_1) \cdot p(l_1) \cdot p(\overline{z}_1 \mid r_1, l_1)$$



Filtering the SLAM DBN: estimation

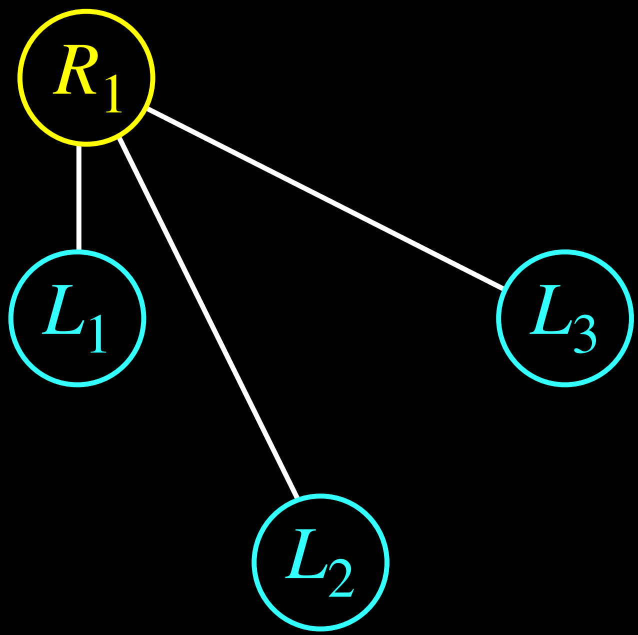
$$p(r_1, l_1 \mid \overline{z}_1) \propto p(r_1) \cdot p(l_1) \cdot p(\overline{z}_1 \mid r_1, l_1)$$



Observing landmark i connects R_t and L_i.

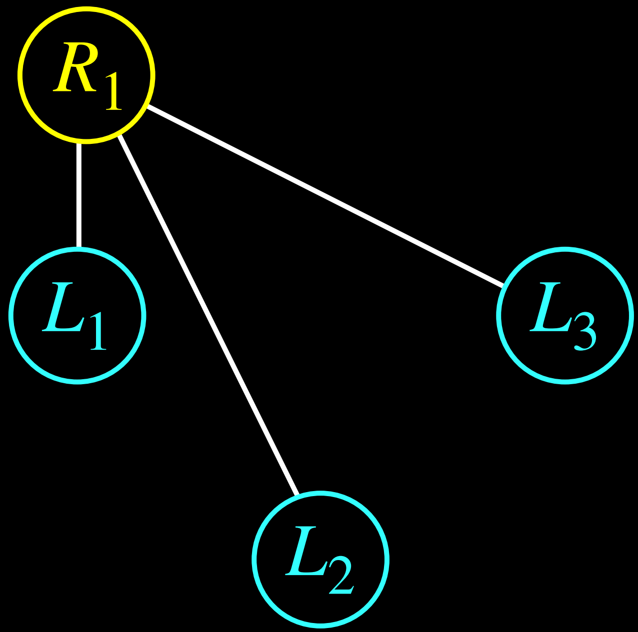
Filtering the SLAM DBN: prediction

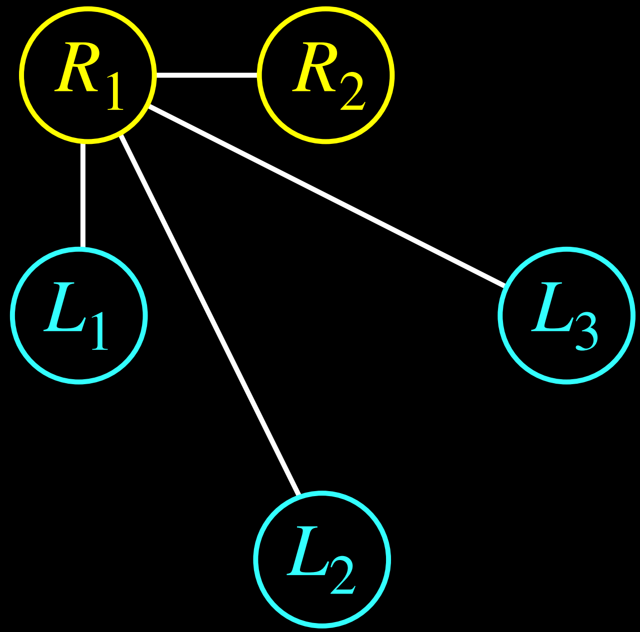
$$p(r_1, l_{1:3} \mid \overline{z}_{1:3})$$



Filtering the SLAM DBN: prediction

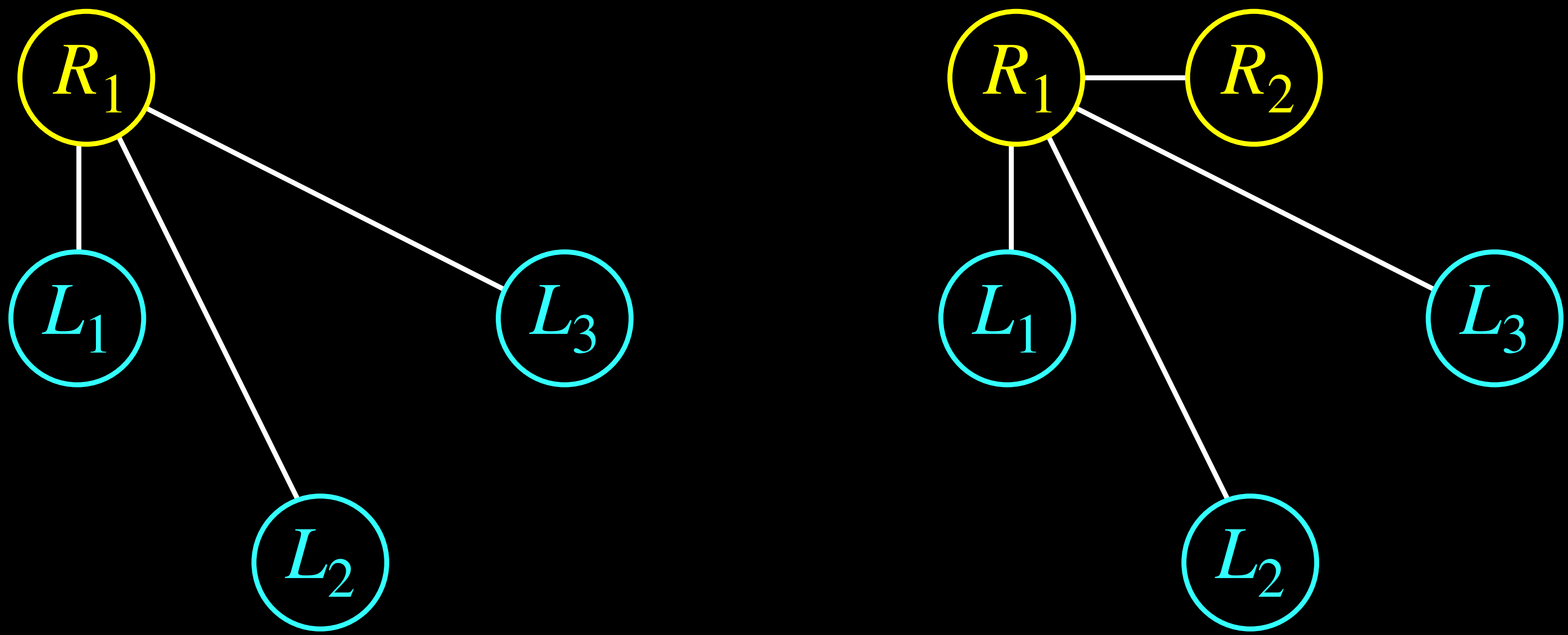
$$p(r_{1:2}, l_{1:3} \mid \overline{z}_{1:3}) = p(r_1, l_{1:3} \mid \overline{z}_{1:3}) \cdot p(r_2 \mid r_1)$$





Filtering the SLAM DBN: prediction

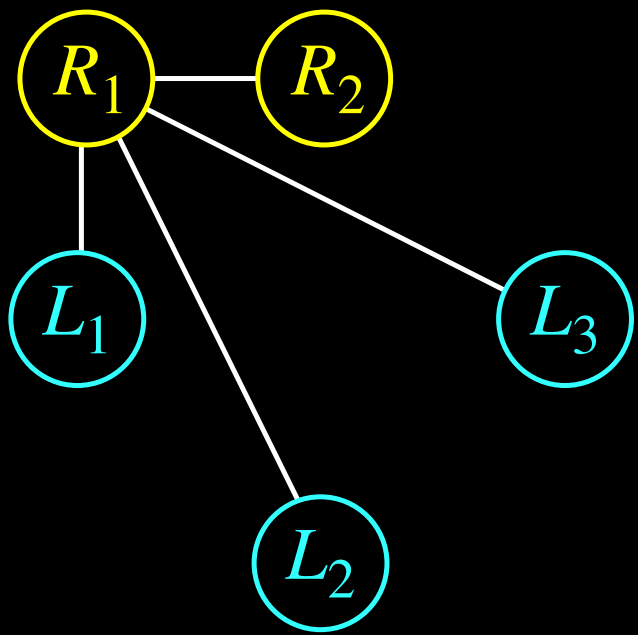
$$p(r_{1:2}, l_{1:3} \mid \overline{z}_{1:3}) = p(r_1, l_{1:3} \mid \overline{z}_{1:3}) \cdot p(r_2 \mid r_1)$$



The prediction phase connects R_t and R_{t+1}.

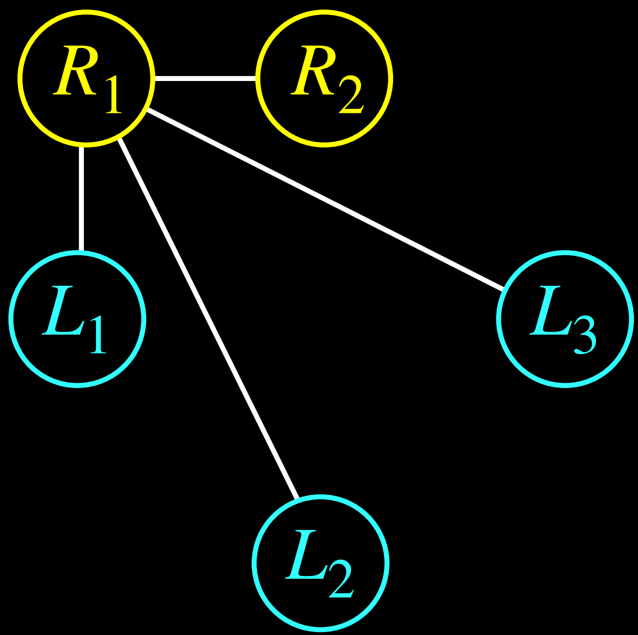
Filtering the SLAM DBN: roll-up

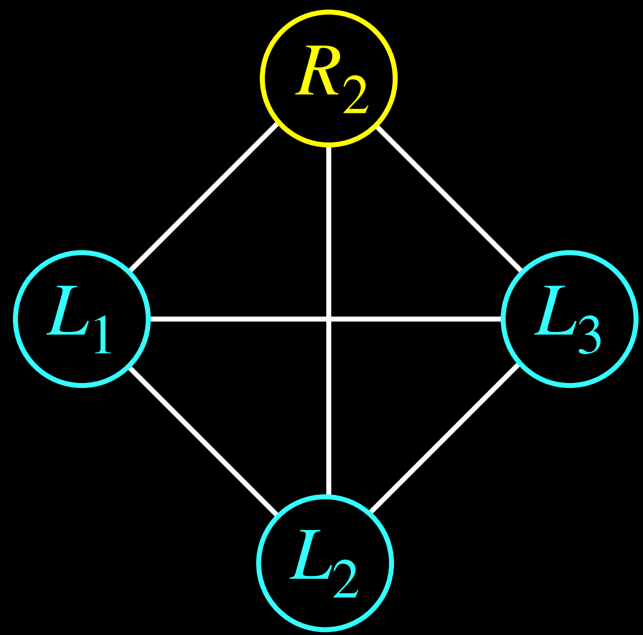
$$p(r_{1:2}, l_{1:3} \mid \overline{z}_{1:3})$$



Filtering the SLAM DBN: roll-up

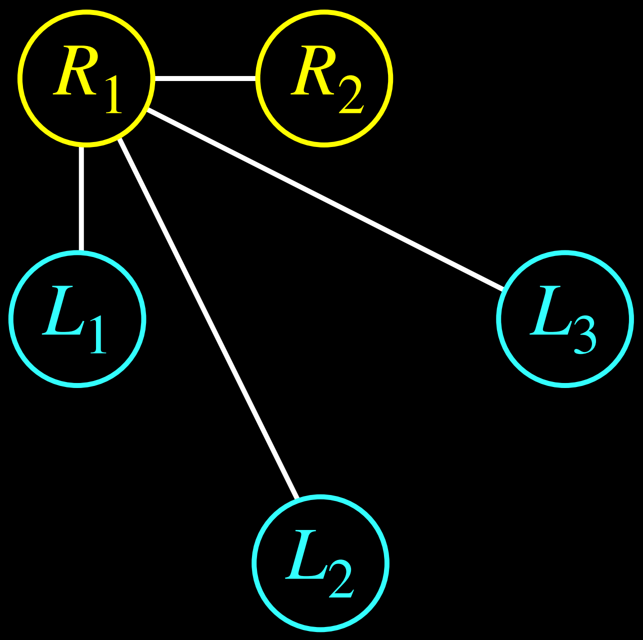
$$p(r_2, l_{1:3} \,|\, \overline{z}_{1:3}) = \sum_{r_1} p(r_{1:2}, l_{1:3} \,|\, \overline{z}_{1:3})$$





Filtering the SLAM DBN: roll-up

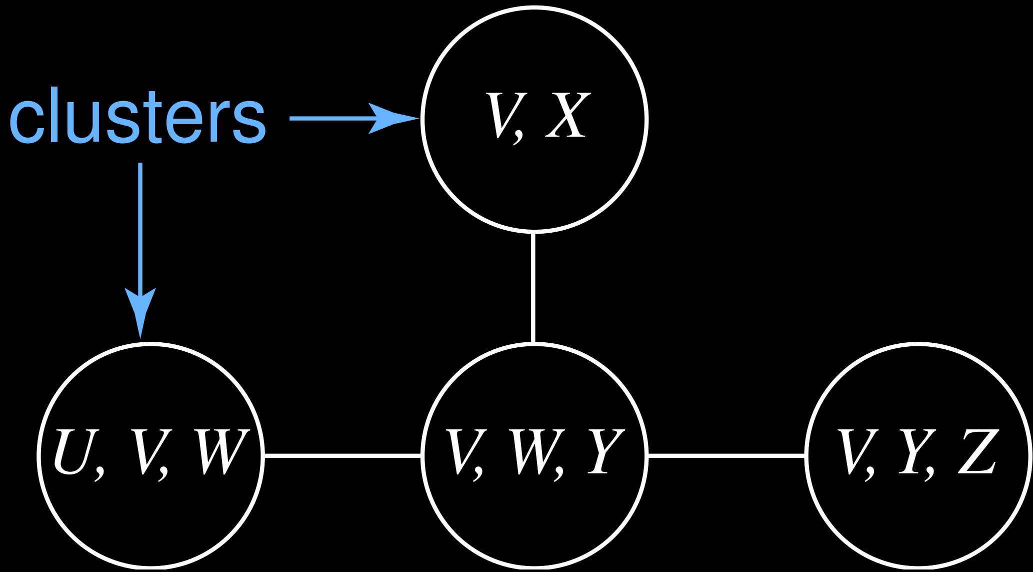
$$p(r_2, l_{1:3} \,|\, \overline{z}_{1:3}) = \sum_{r_1} p(r_{1:2}, l_{1:3} \,|\, \overline{z}_{1:3})$$



After roll-up, the SLAM belief state has no conditional independencies.

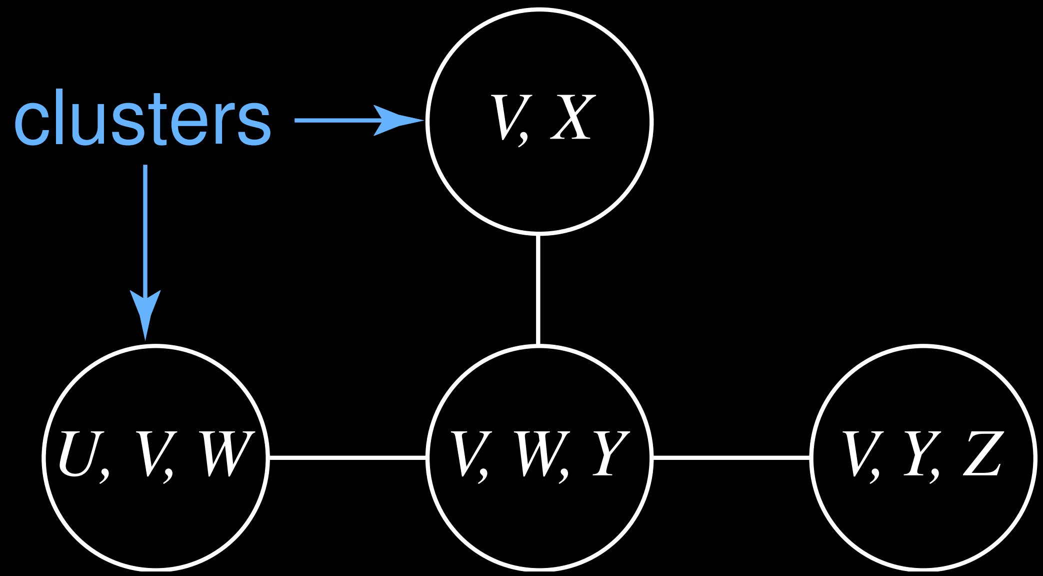
Junction trees

An undirected tree whose nodes are sets of variables. . .



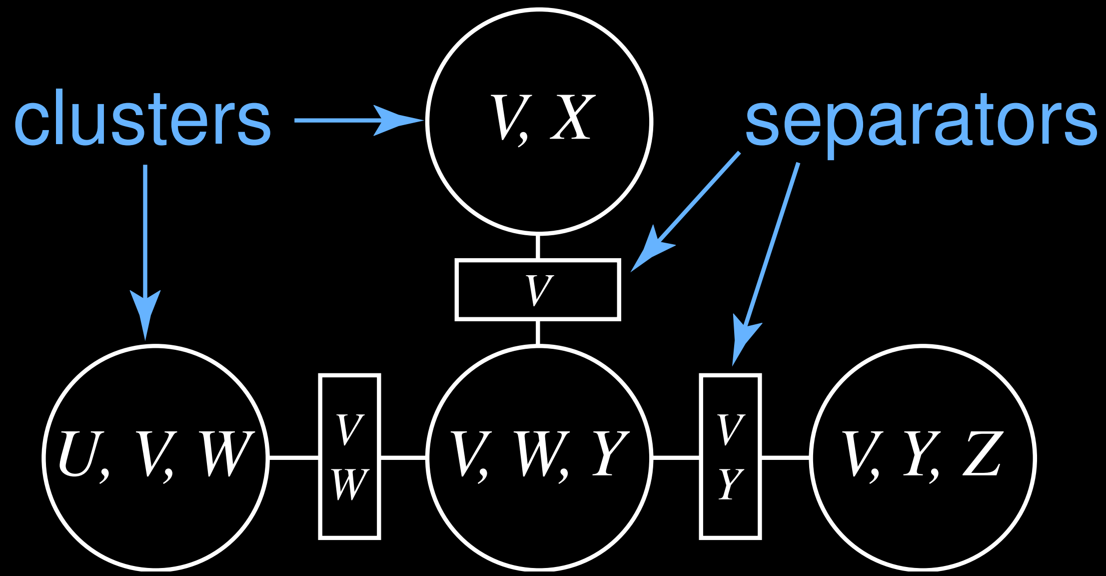
Junction trees

An undirected tree whose nodes are sets of variables. . . with the running intersection property.



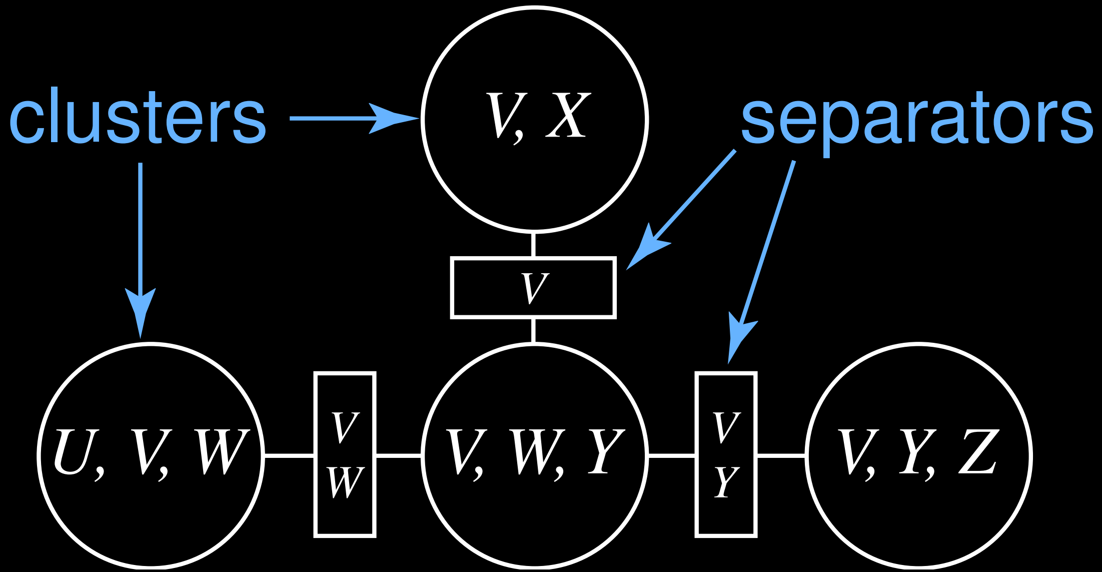
Junction trees

An undirected tree whose nodes are sets of variables. . . with the running intersection property.



Junction trees

An undirected tree whose nodes are sets of variables. . . with the running intersection property.

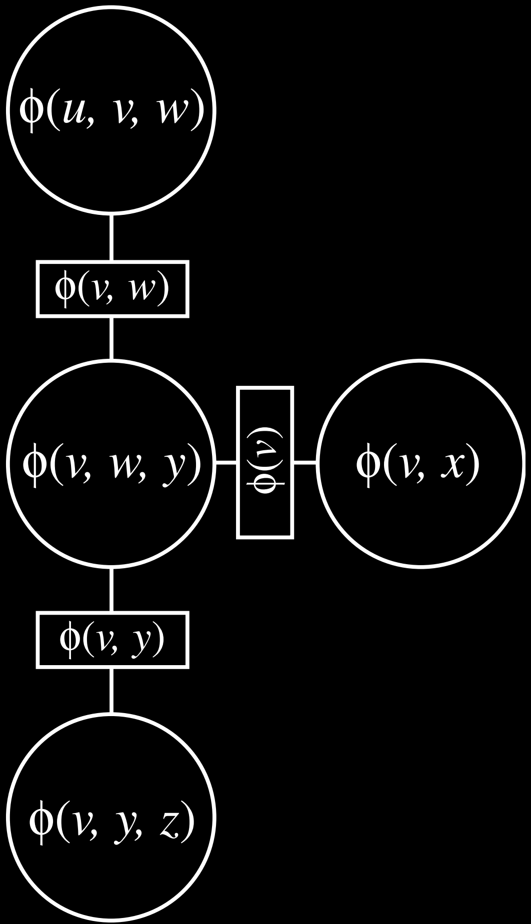


A density p decomposes on T if we can write

$$p(x) = \frac{\prod_C \phi_C(x_C)}{\prod_S \phi_S(x_S)}$$

Junction tree inference

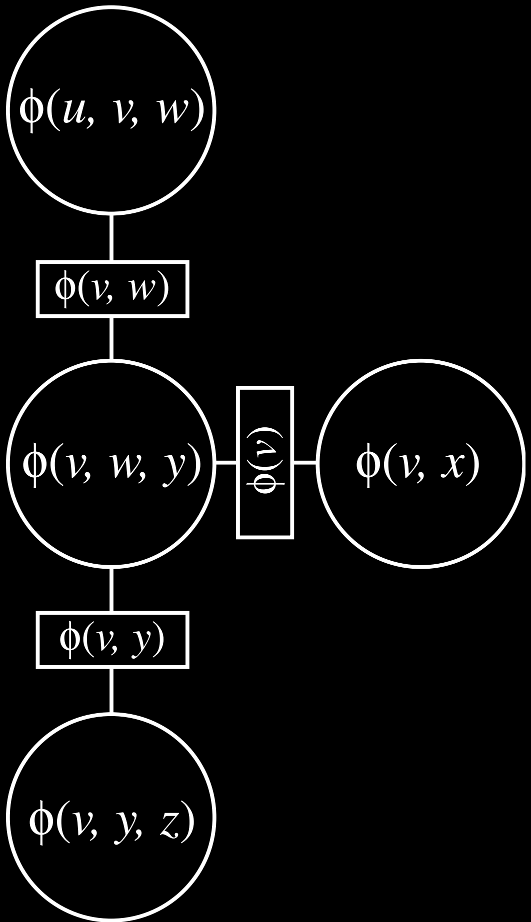
initialize



$$p = \frac{\prod_C \phi_C}{\prod_S \phi_S}$$

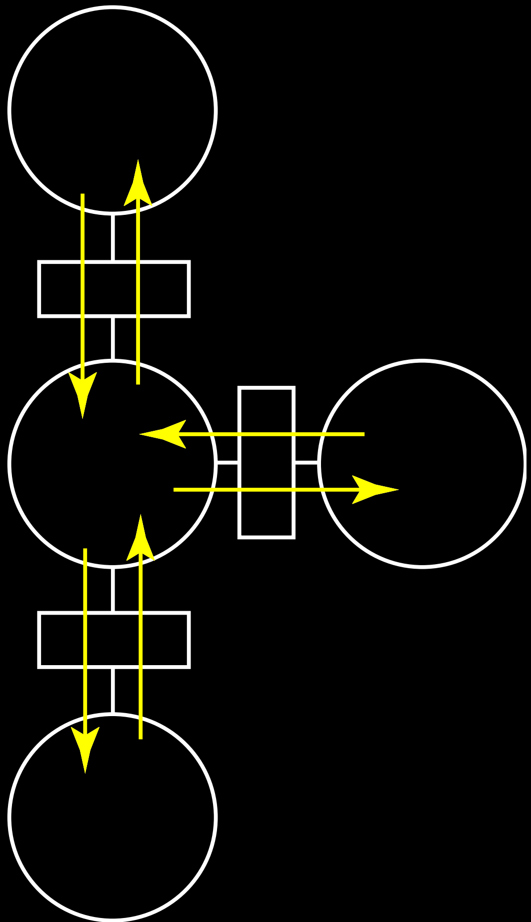
Junction tree inference

initialize



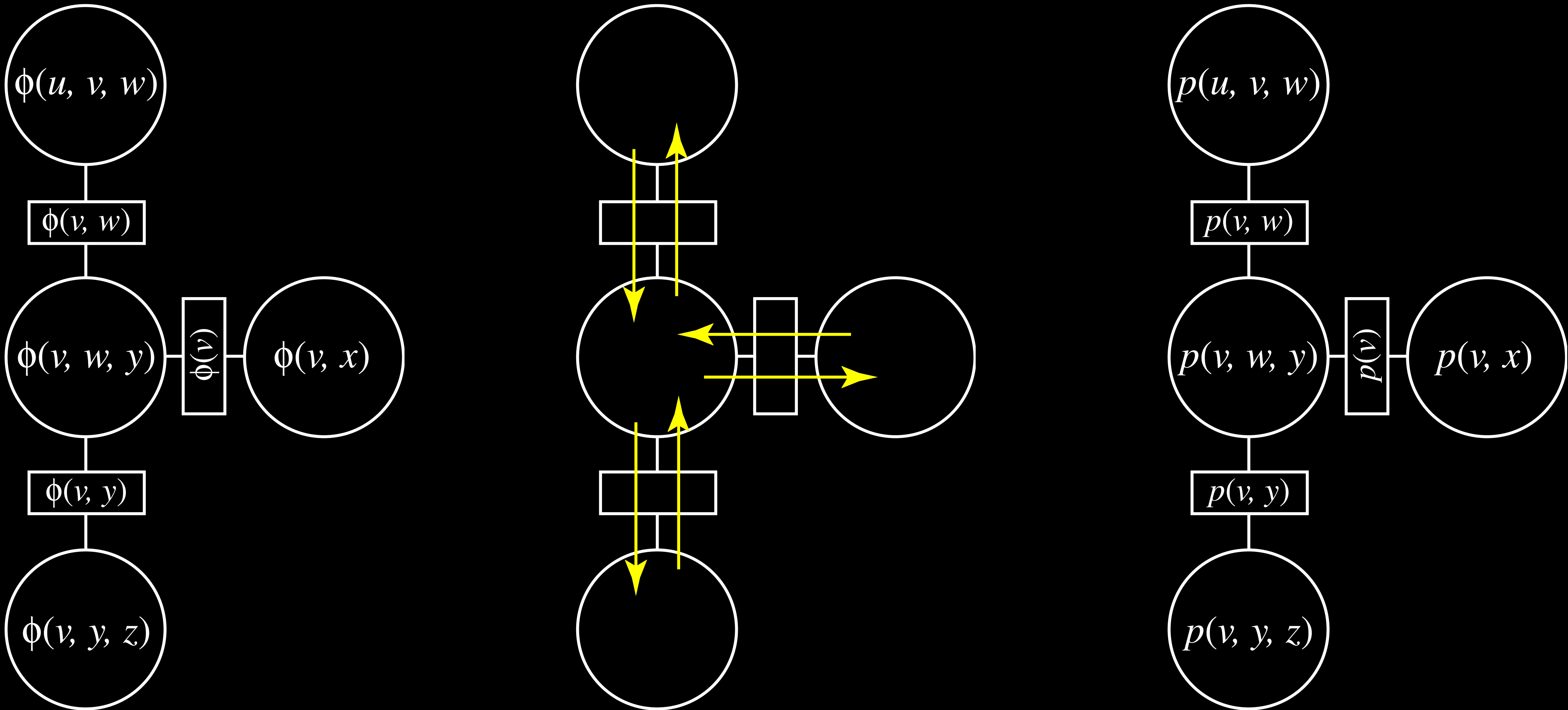
$$p = \frac{\prod_C \phi_C}{\prod_S \phi_S}$$

pass messages



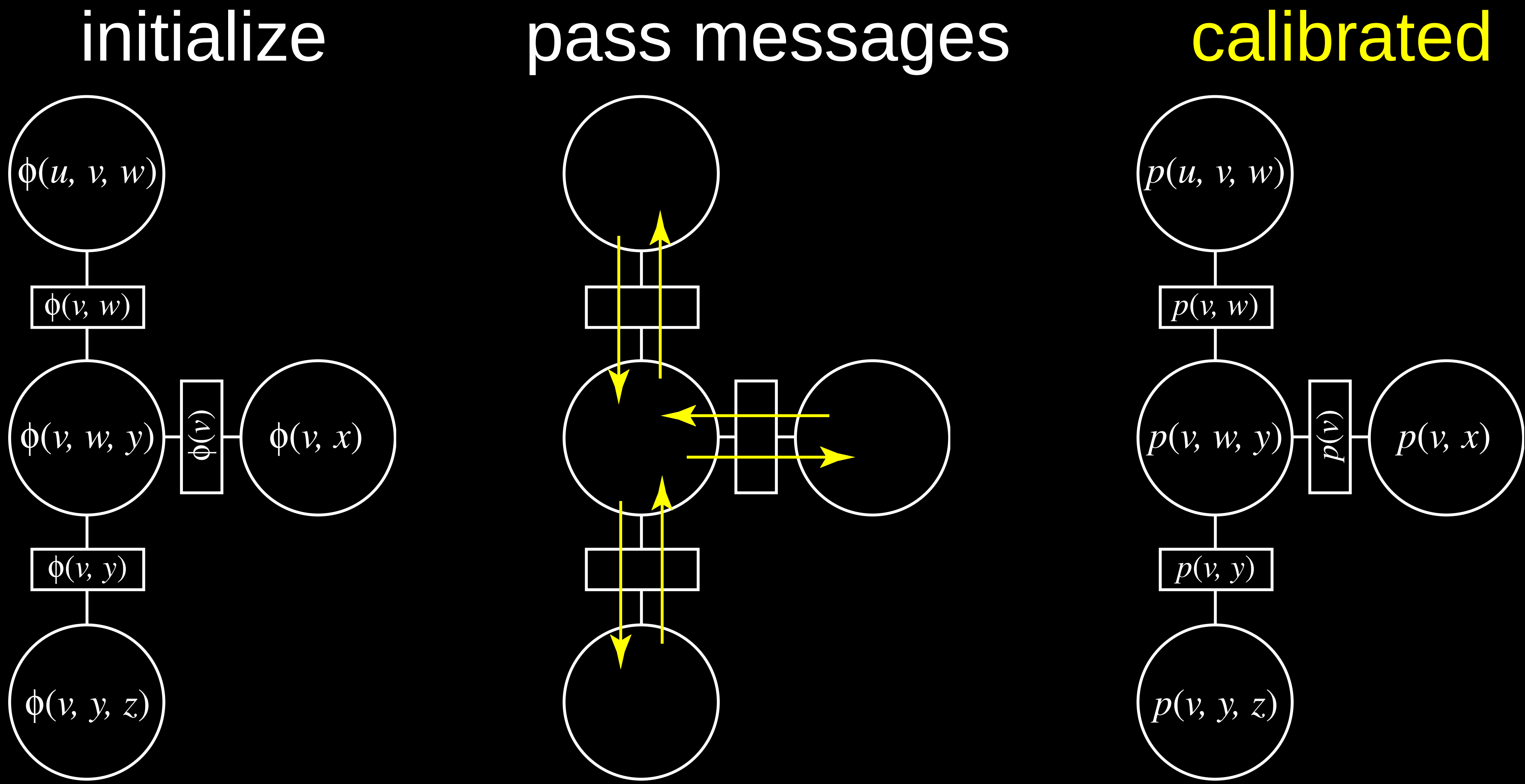
Junction tree inference

initialize | pass messages | calibrated



$$p = \frac{\prod_C \phi_C}{\prod_S \phi_S}$$

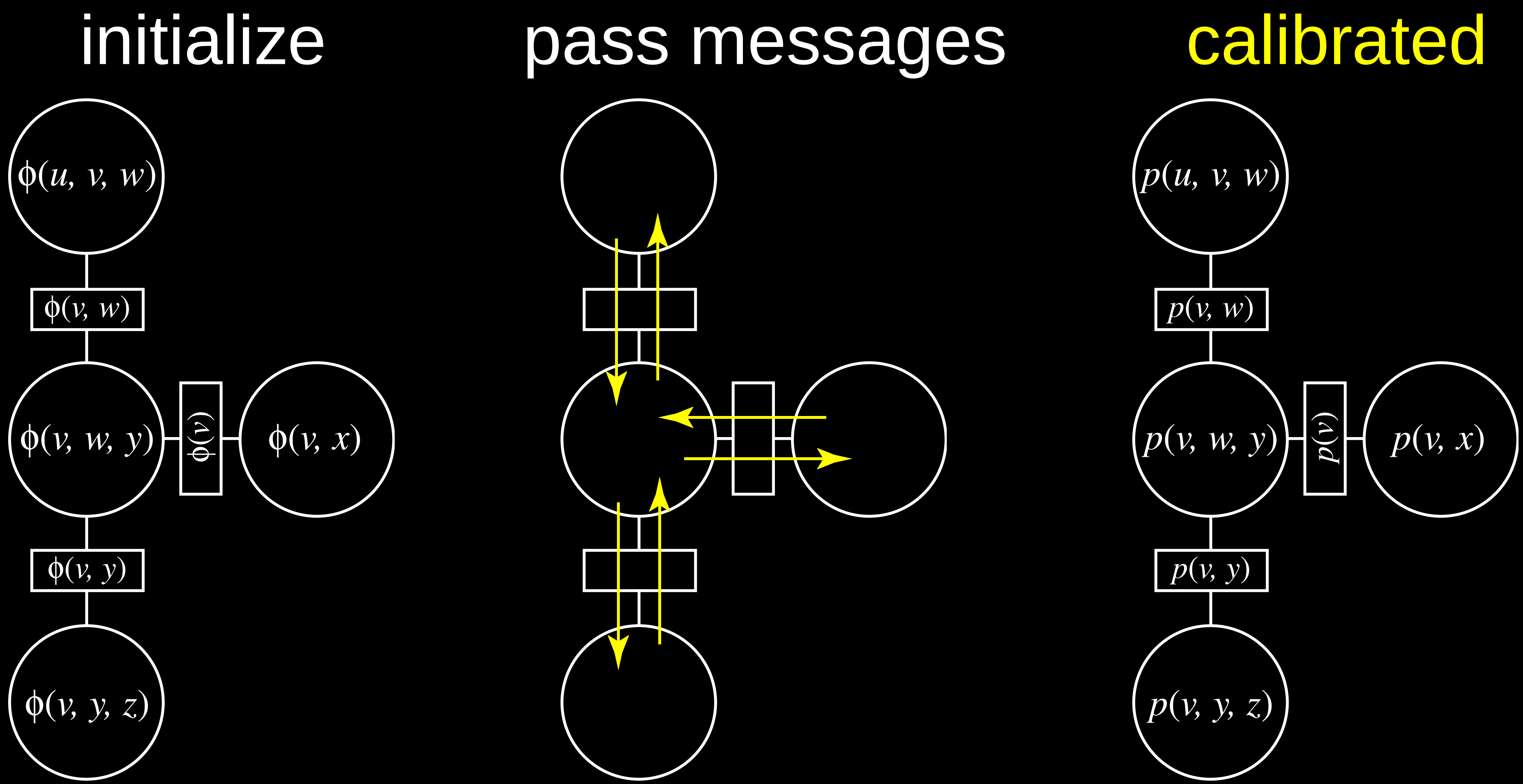
Junction tree inference



$$p = \frac{\prod_C \phi_C}{\prod_S \phi_S}$$

$$p = \frac{\prod_C p_C}{\prod_S p_S}$$

Junction tree inference



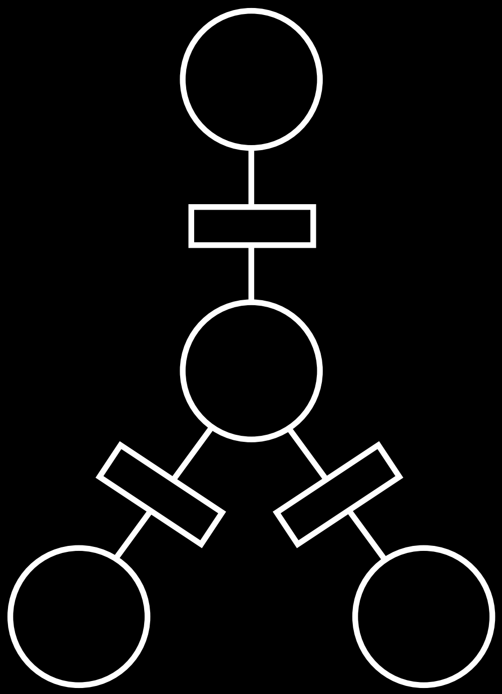
$$p = \frac{\prod_C \phi_C}{\prod_S \phi_S}$$

complexity scales with width

$$p = \frac{\prod_C p_C}{\prod_S p_S}$$

Junction tree filters

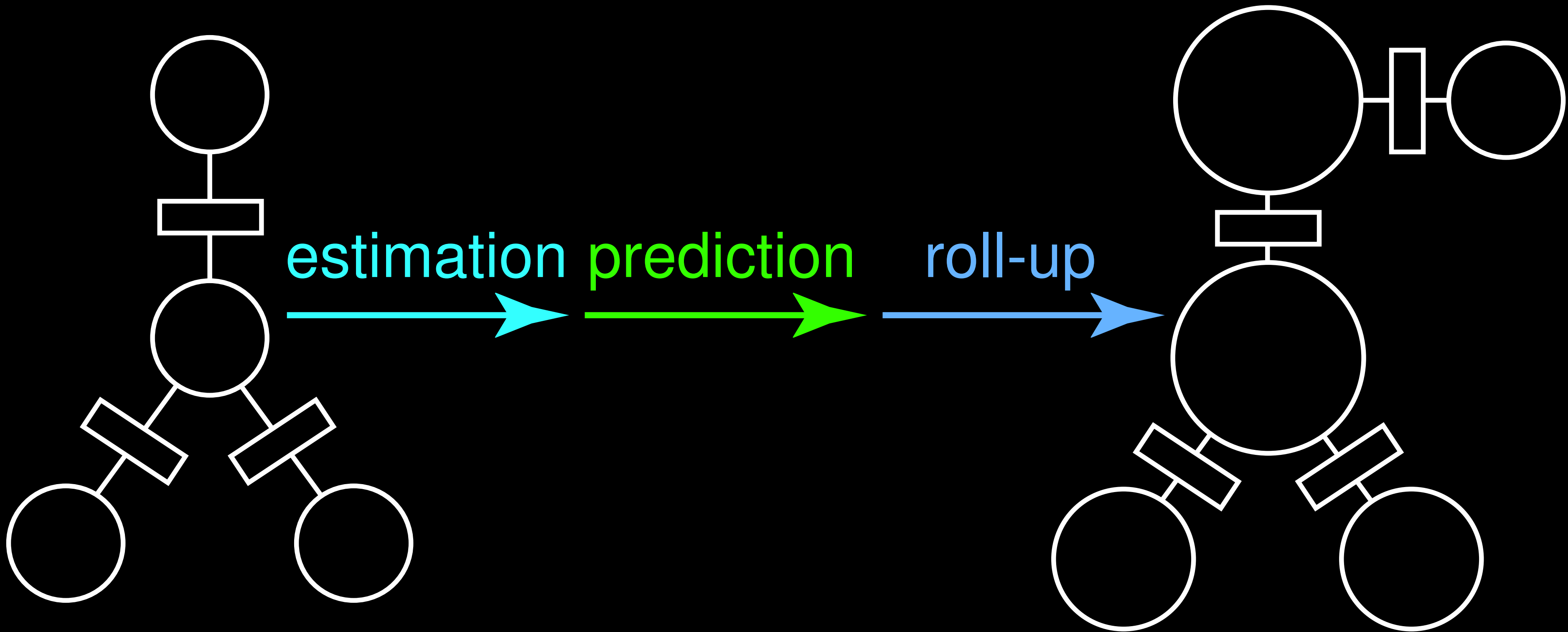
The belief state is a calibrated junction tree.



$$p^{(t)} = \frac{\prod_C p_C^{(t)}}{\prod_S p_S^{(t)}}$$

Junction tree filters

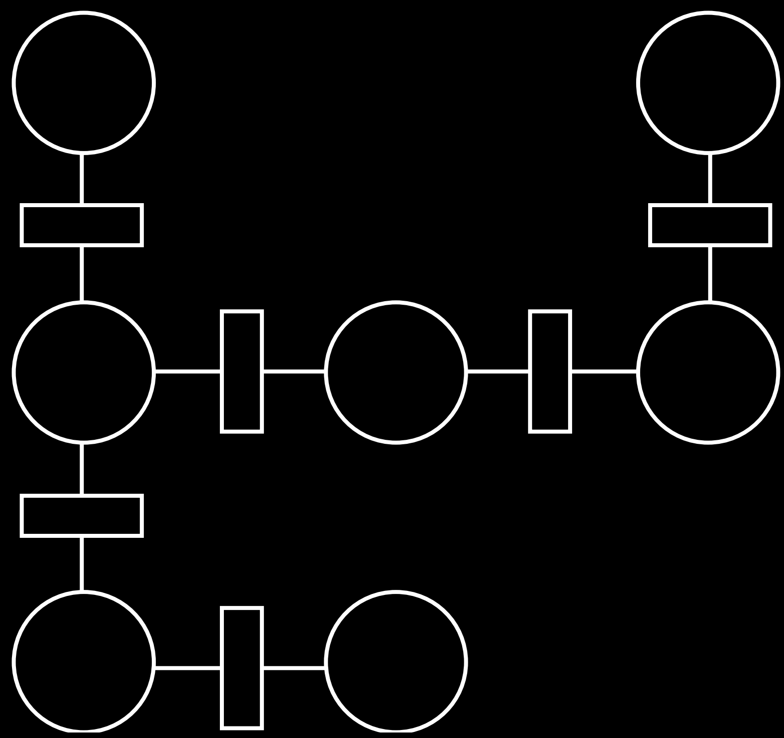
The belief state is a calibrated junction tree.



$$p^{(t)} = \frac{\prod_C p_C^{(t)}}{\prod_S p_S^{(t)}} \qquad p^{(t+1)} = \frac{\prod_C p_C^{(t+1)}}{\prod_S p_S^{(t+1)}}$$

Estimation and prediction

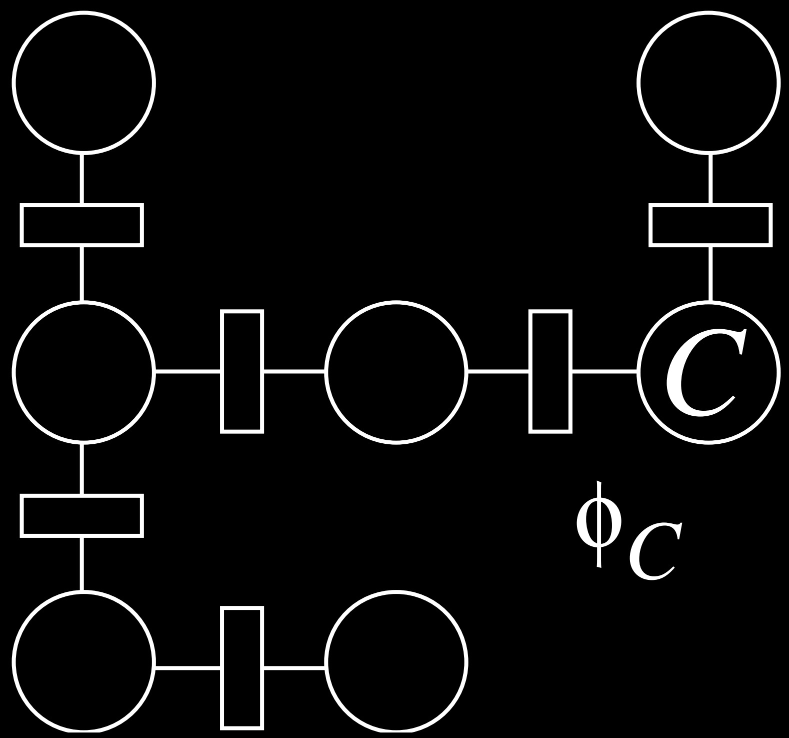
To multiply $\psi(x_1, \ldots, x_k)$ into p and recalibrate:



Estimation and prediction

To multiply $\psi(x_1, \ldots, x_k)$ into p and recalibrate:

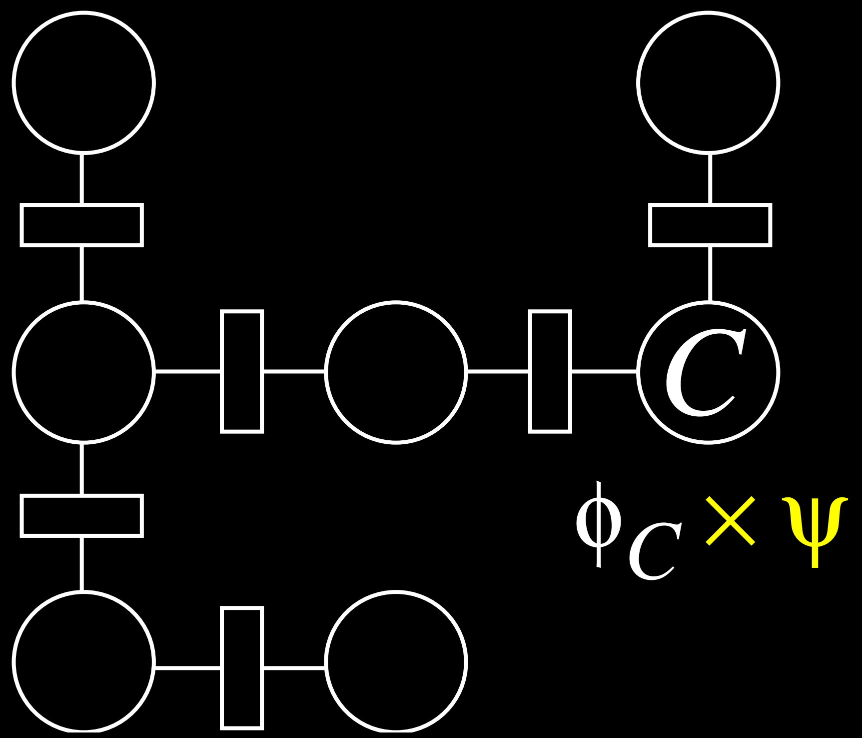
1. Find a cluster C that contains $X_1, \ldots, X_k$.



Estimation and prediction

To multiply $\psi(x_1, \ldots, x_k)$ into p and recalibrate:

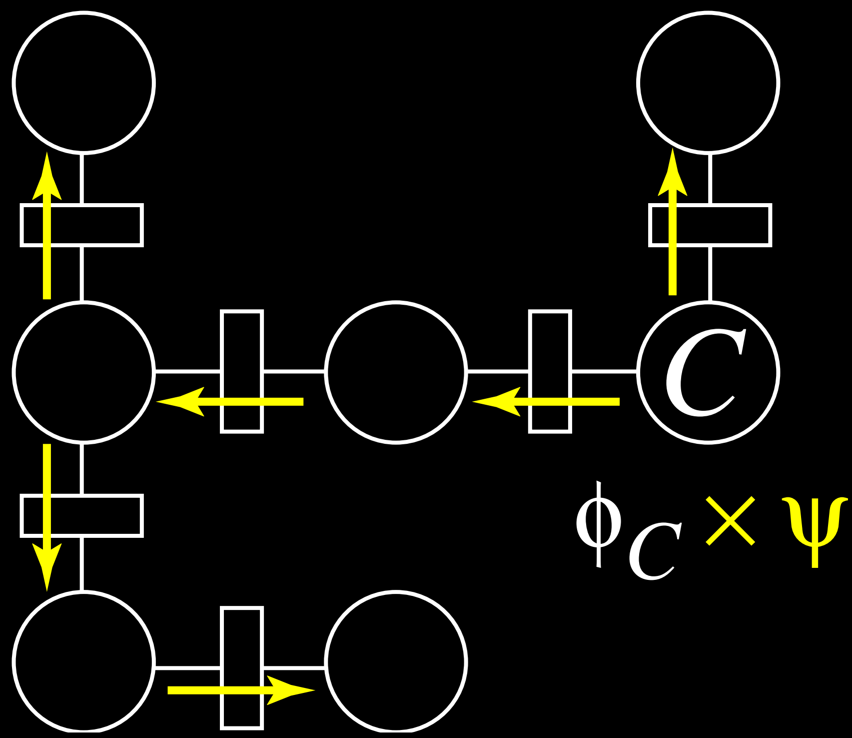
1. Find a cluster C that contains $X_1, \ldots, X_k$.
2. Multiply ψ into ϕ_C.



Estimation and prediction

To multiply $\psi(x_1, \ldots, x_k)$ into p and recalibrate:

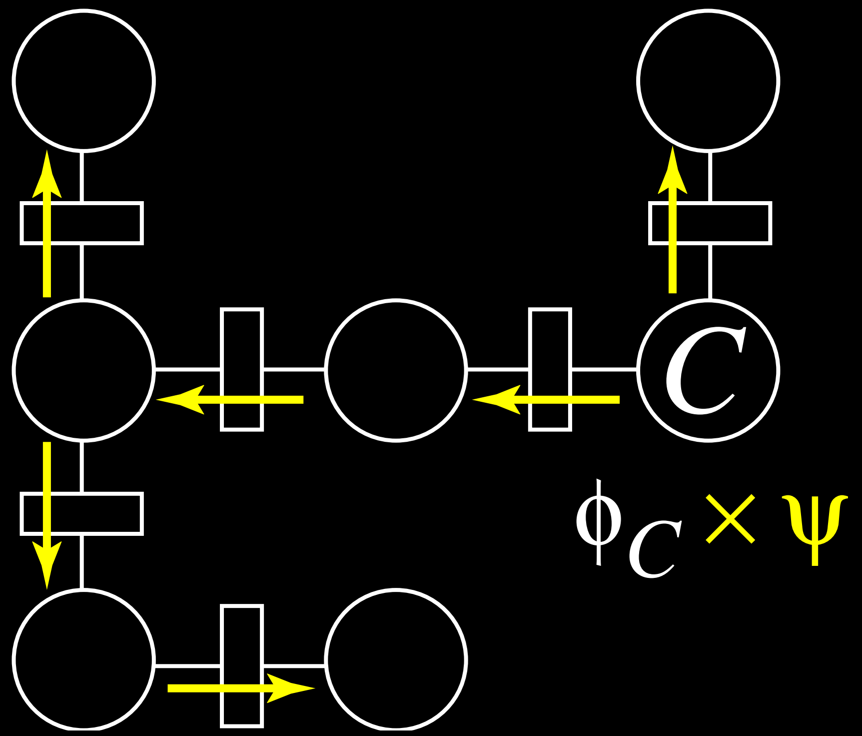
1. Find a cluster C that contains $X_1, \ldots, X_k$.
2. Multiply ψ into ϕ_C.
3. Distribute evidence from C (if needed).



Estimation and prediction

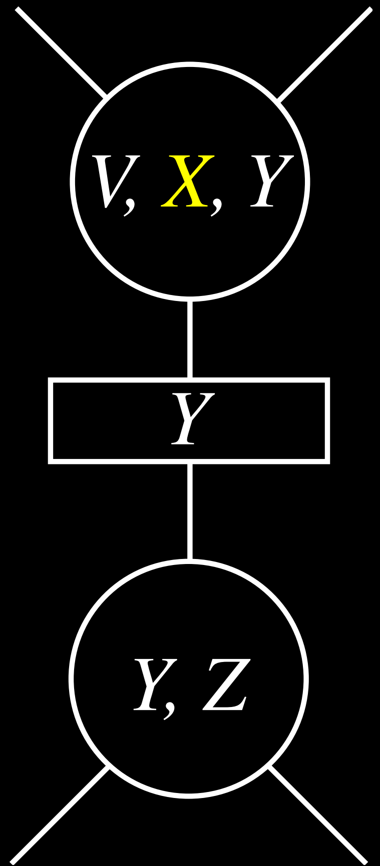
To multiply $\psi(x_1, \ldots, x_k)$ into p and recalibrate:

1. Find a cluster C that contains $X_1, \ldots, X_k$.
2. Multiply ψ into ϕ_C.
3. Distribute evidence from C (if needed).

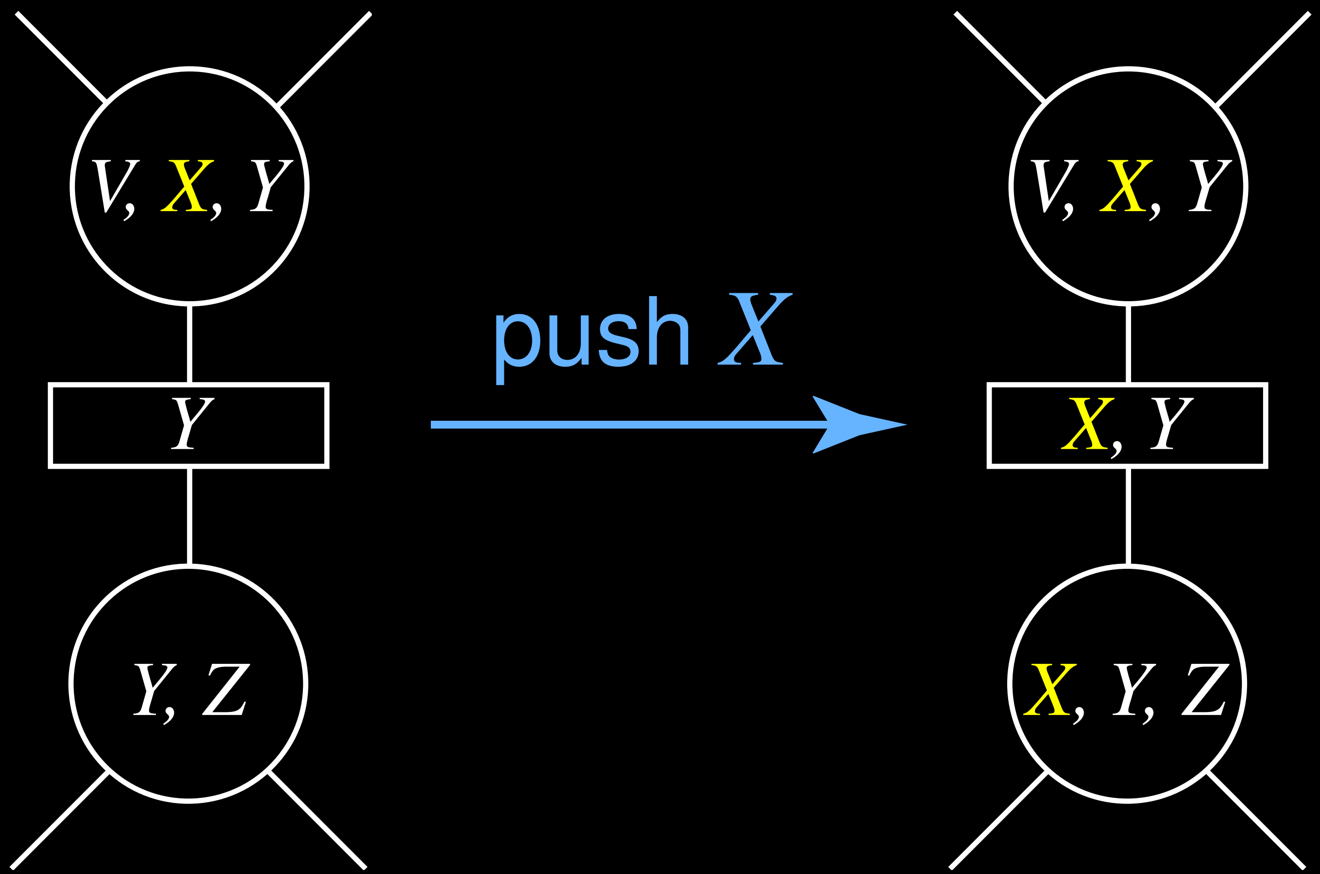


If there is no cover, we must make one.

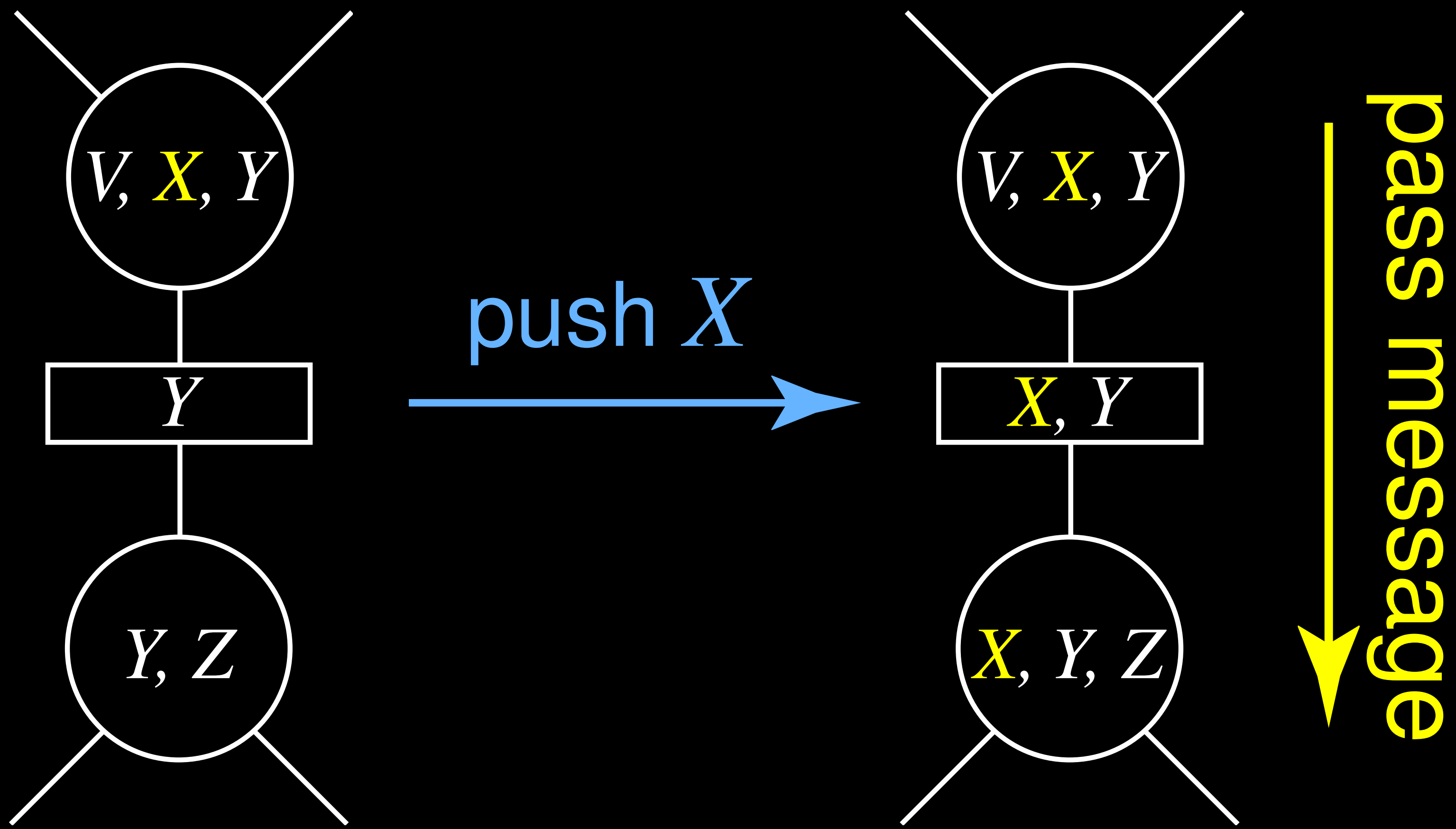
Pushing variables to create covers



Pushing variables to create covers

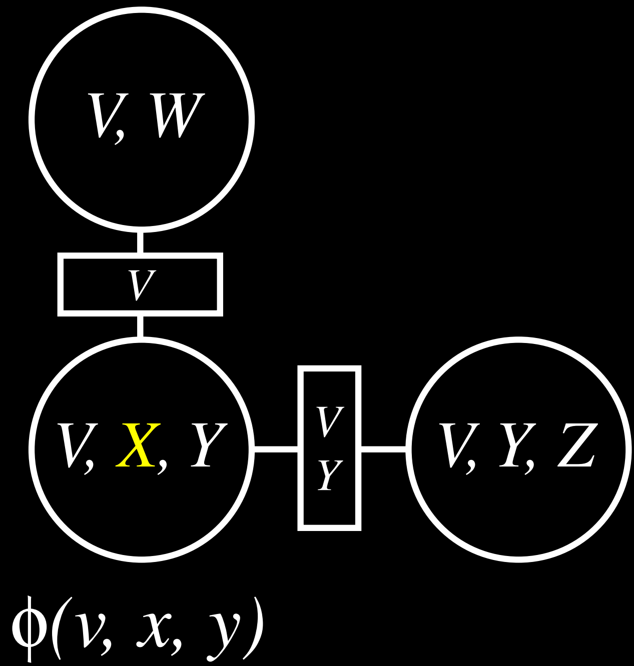


Pushing variables to create covers



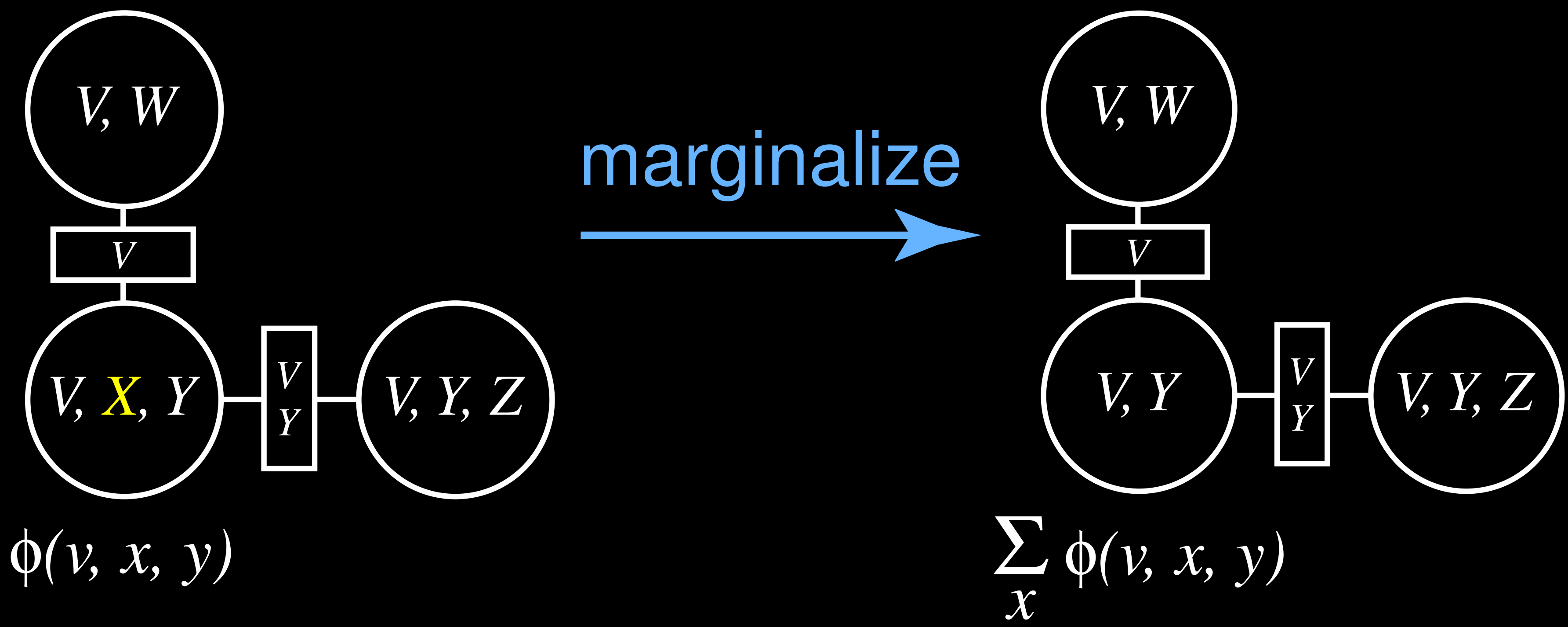
Roll-up

To marginalize out a variable X that is in only one cluster C . . .



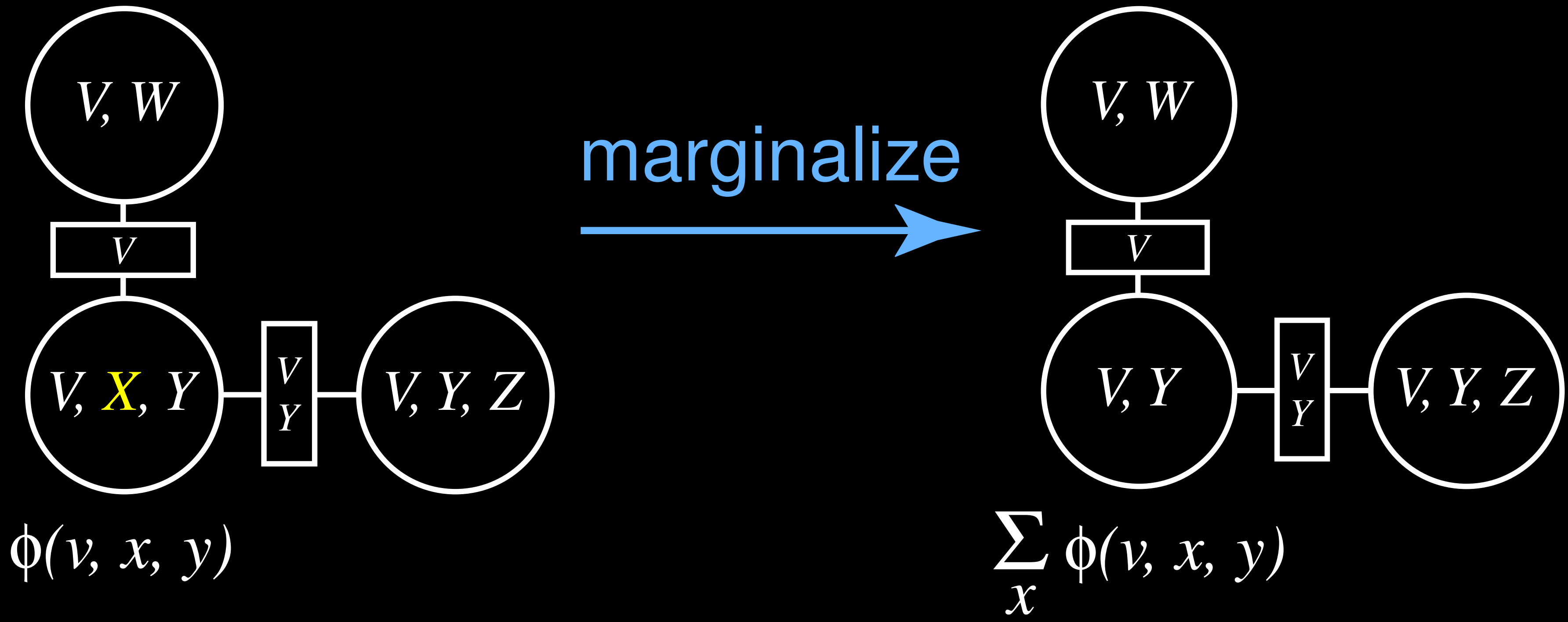
Roll-up

To marginalize out a variable X that is in only one cluster C . . . marginalize X out of ϕ_C.



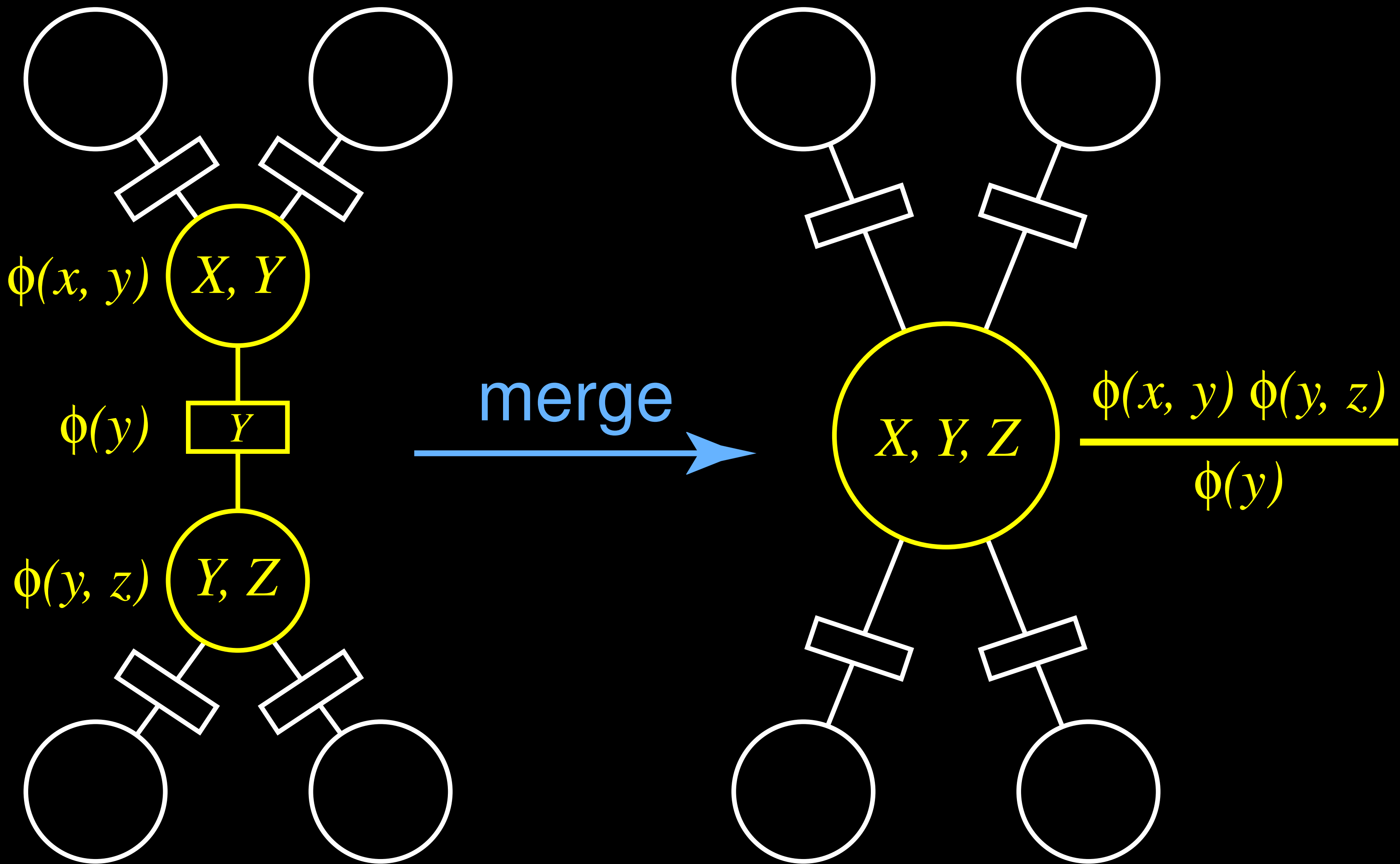
Roll-up

To marginalize out a variable X that is in only one cluster C ... marginalize X out of ϕ_C.



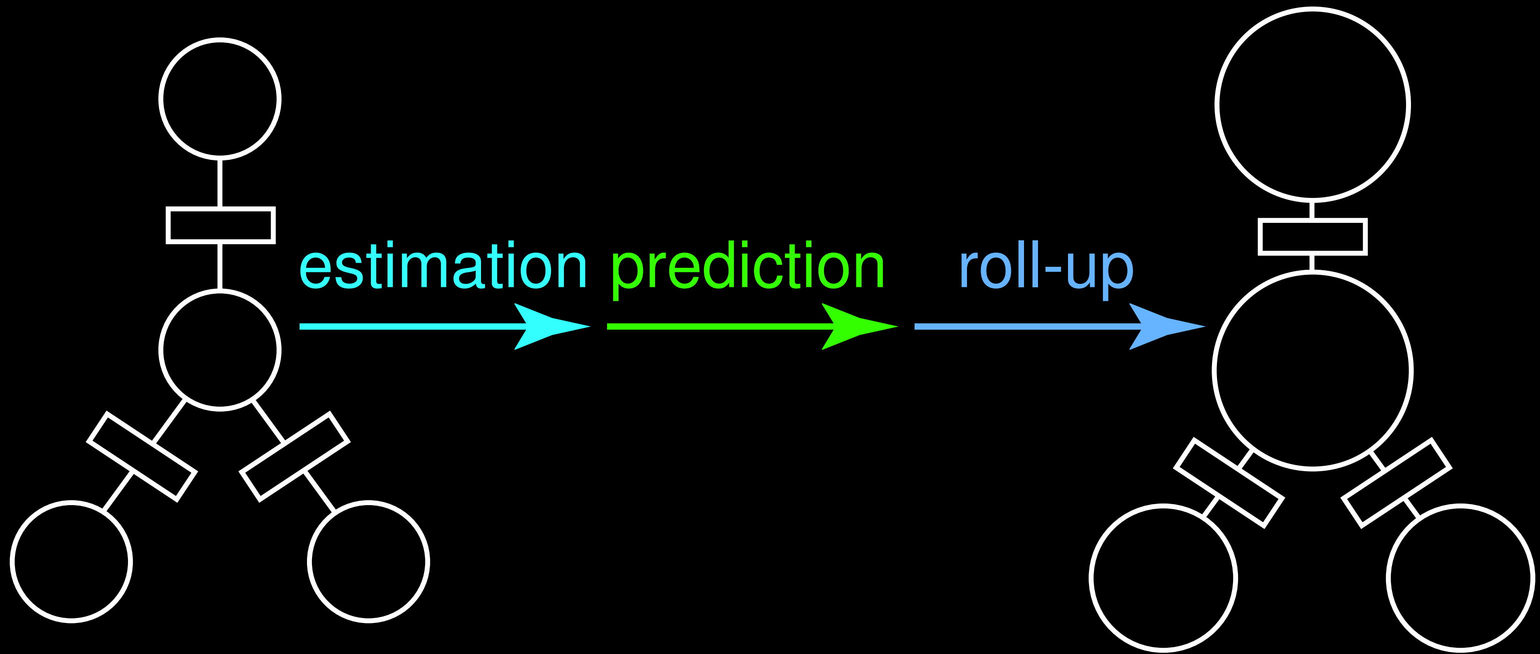
If X is in more than one cluster, we must first merge the clusters containing X...

Merging adjacent clusters



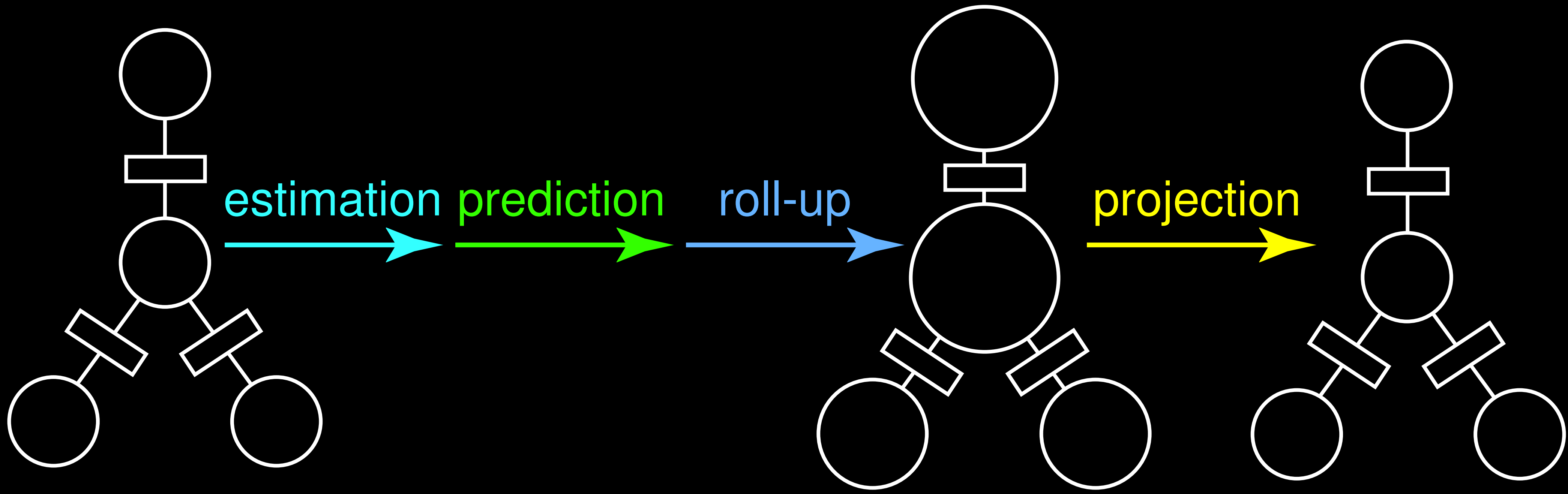
Thin junction tree filters (TJTF)

Pushing and merging increase the width of the junction tree, and therefore the complexity.



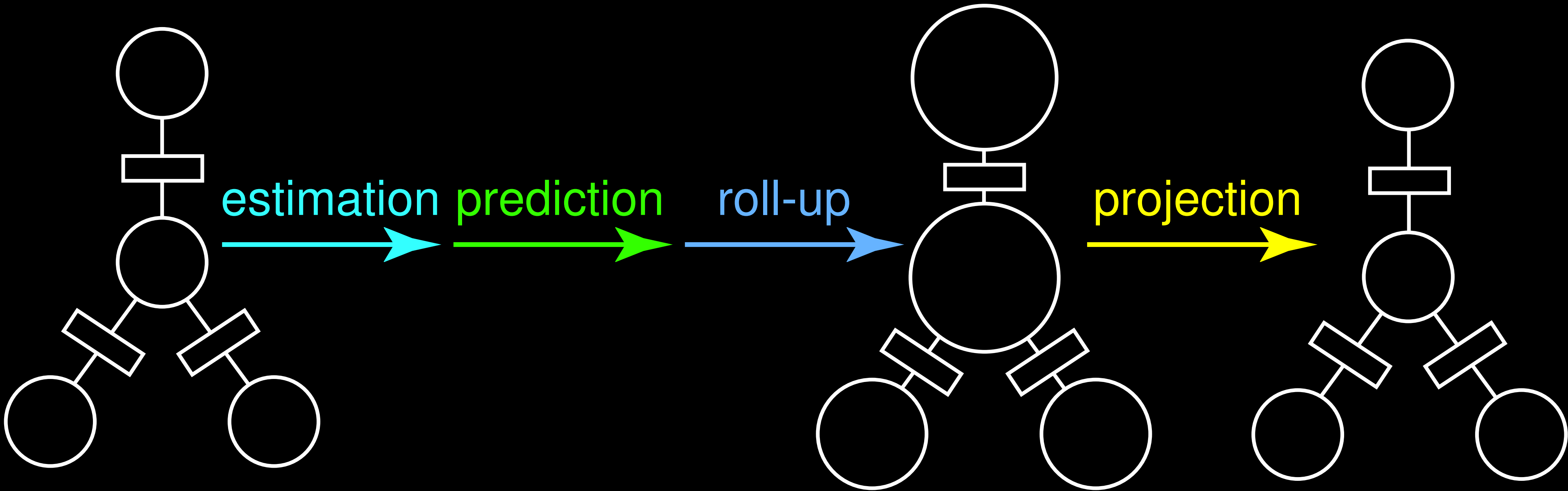
Thin junction tree filters (TJTF)

Pushing and merging increase the width of the junction tree, and therefore the complexity.



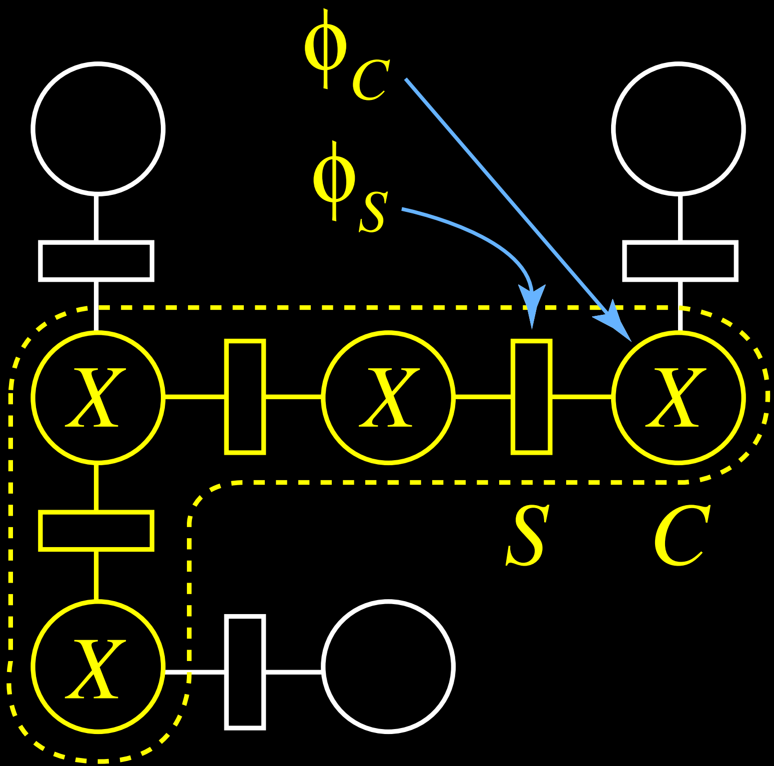
Thin junction tree filters (TJTF)

Pushing and merging increase the width of the junction tree, and therefore the complexity.

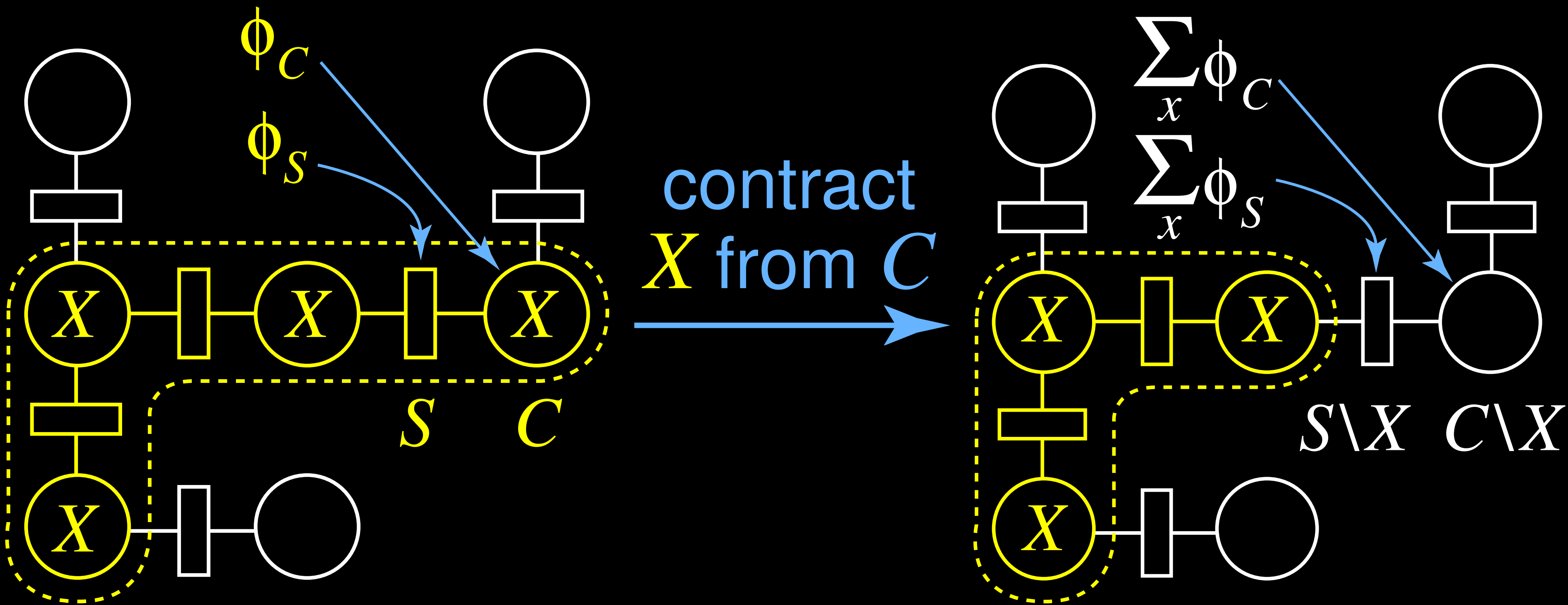


TJTF chooses the projection adaptively to minimize the approximation error.

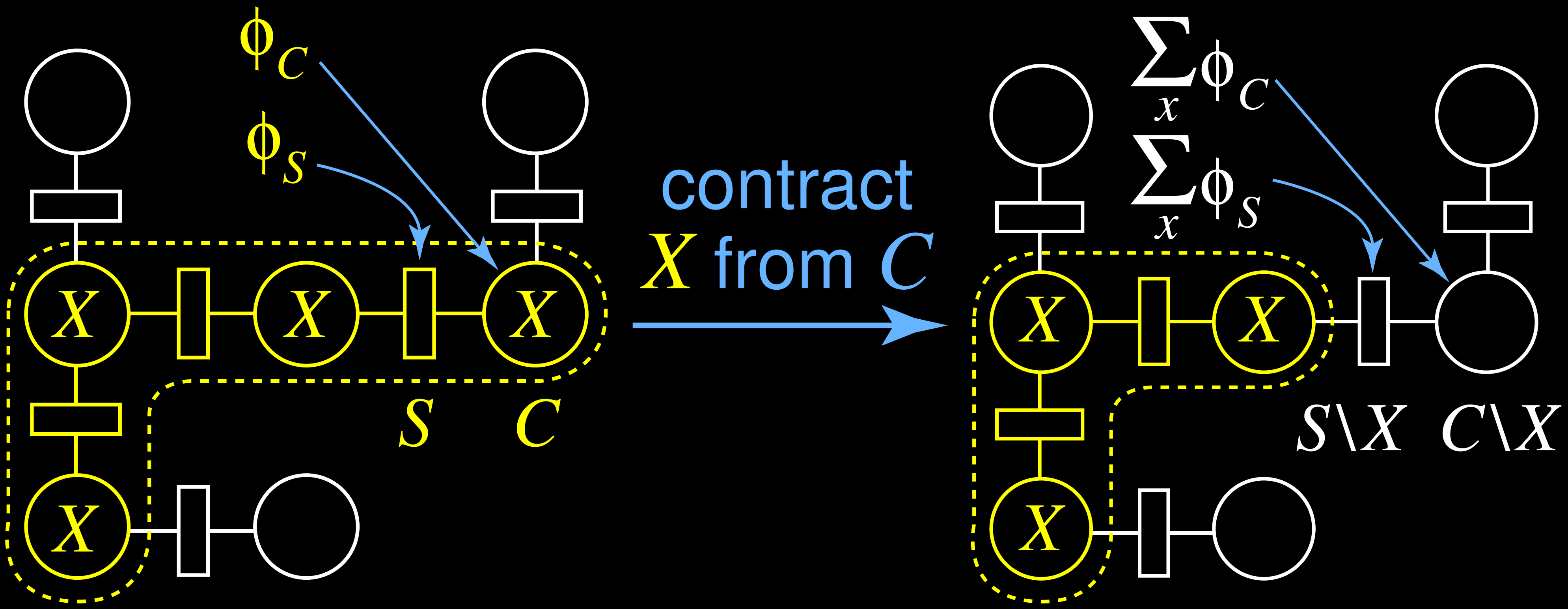
Variable contraction



Variable contraction



Variable contraction



This cuts all edges between X and $C - S$, the variables X no longer resides with.

Variable contraction is an I-projection

Proposition. If $\tilde{p}$ is the density obtained by contracting X from C, then

$$\tilde{p} = \underset{\{q \,:\, X \perp\!\!\!\perp (C - S) \mid (S \setminus X)\}}{\arg\min} D(p \,||\, q)$$

Adaptive approximation

Proposition. If $\tilde{p}$ is the density obtained by contracting X from C, then

$$D(p \,||\, \tilde{p}) = I(X; C - S \,|\, S \setminus X)$$

Adaptive approximation

Proposition. If $\tilde{p}$ is the density obtained by contracting X from C, then

$$D(p \,||\, \tilde{p}) = I(X; C - S \,|\, S \setminus X)$$

- This can be computed using $\phi_C \propto p_C$.

Adaptive approximation

Proposition. If $\tilde{p}$ is the density obtained by contracting X from C, then

$$D(p \,||\, \tilde{p}) = I(X; C - S \,|\, S \setminus X)$$

- This can be computed using $\phi_C \propto p_C$.
- For Gaussian p, this is $O(\dim(X)^3)$ time!

Adaptive approximation

Proposition. If $\tilde{p}$ is the density obtained by contracting X from C, then

$$D(p \,||\, \tilde{p}) = I(X; C - S \,|\, S \setminus X)$$

- This can be computed using $\phi_C \propto p_C$.
- For Gaussian p, this is $O(\mathrm{dim}(X)^3)$ time!
- To thin C, perform the contraction that minimizes this approximation error.

Thin junction tree filters for SLAM

- The junction tree has $O(N_t)$ clusters.

Thin junction tree filters for SLAM

- The junction tree has $O(N_t)$ clusters.
- Use greedy-optimal variable contractions to keep the width bounded by w.

Thin junction tree filters for SLAM

- The junction tree has $O(N_t)$ clusters.
- Use greedy-optimal variable contractions to keep the width bounded by w.
- Space complexity: $O(w^2 \cdot N_t)$

Thin junction tree filters for SLAM

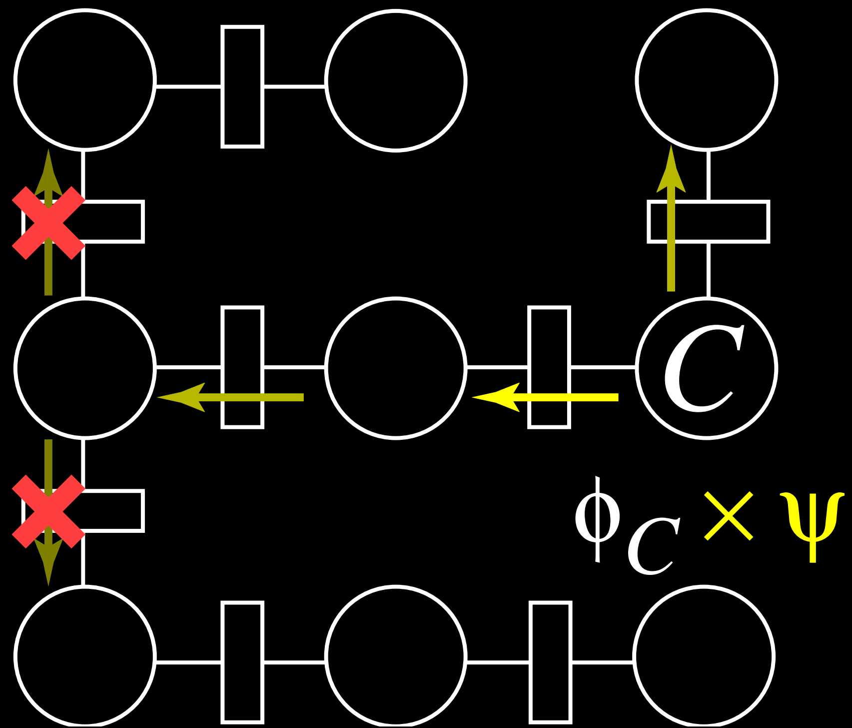
- The junction tree has $O(N_t)$ clusters.
- Use greedy-optimal variable contractions to keep the width bounded by w.
- Space complexity: $O(w^2 \cdot N_t)$
- Time complexity: $O(w^3 \cdot N_t)$

Thin junction tree filters for SLAM

- The junction tree has $O(N_t)$ clusters.
- Use greedy-optimal variable contractions to keep the width bounded by w.
- Space complexity: $O(w^2 \cdot N_t)$
- Time complexity: $O(w^3 \cdot N_t)$
- This $O(N_t)$ time complexity is due (mainly) to message passing in the estimation step.

Adaptive message passing

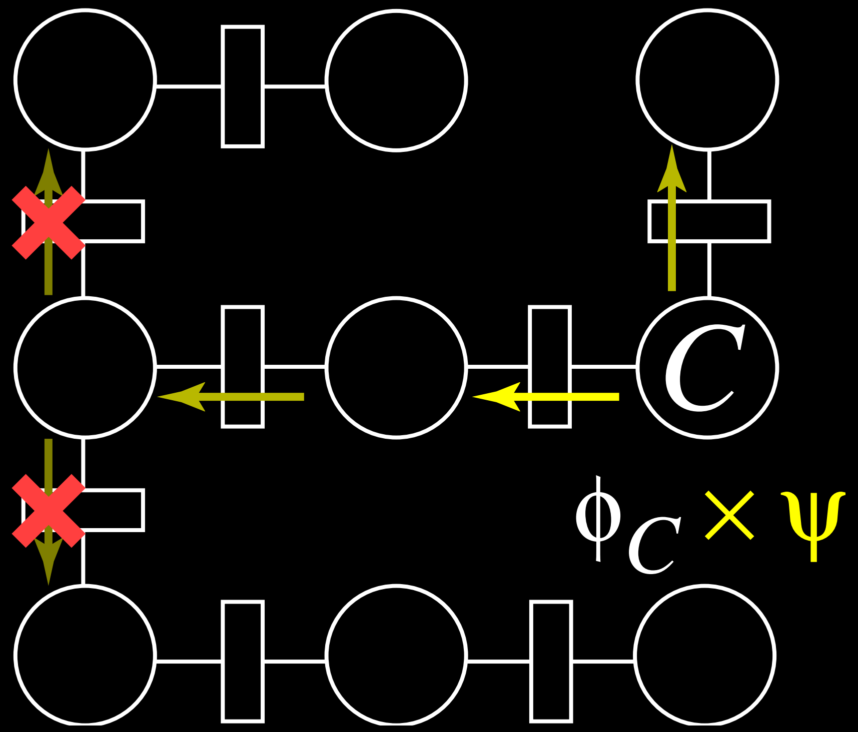
Propagate messages only as long as they induce significant change in the belief state.



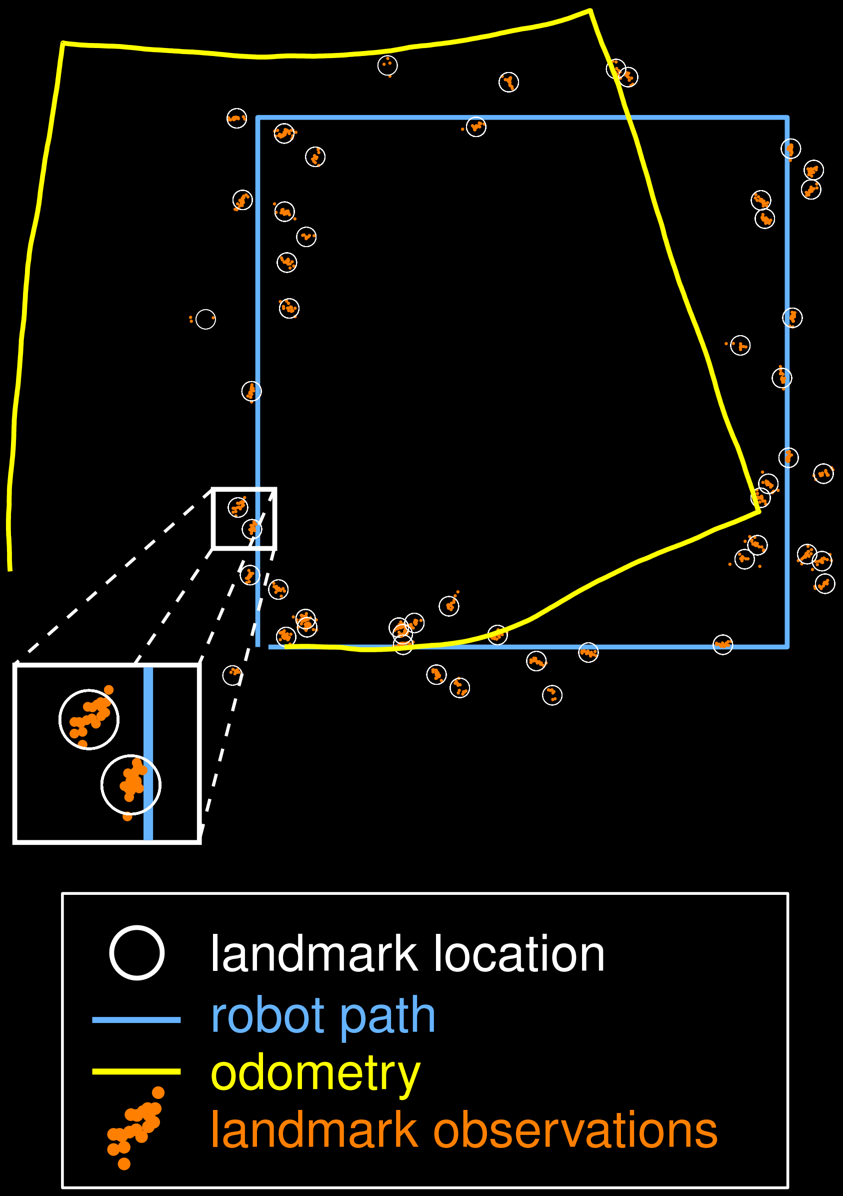
Adaptive message passing

Propagate messages only as long as they induce significant change in the belief state.

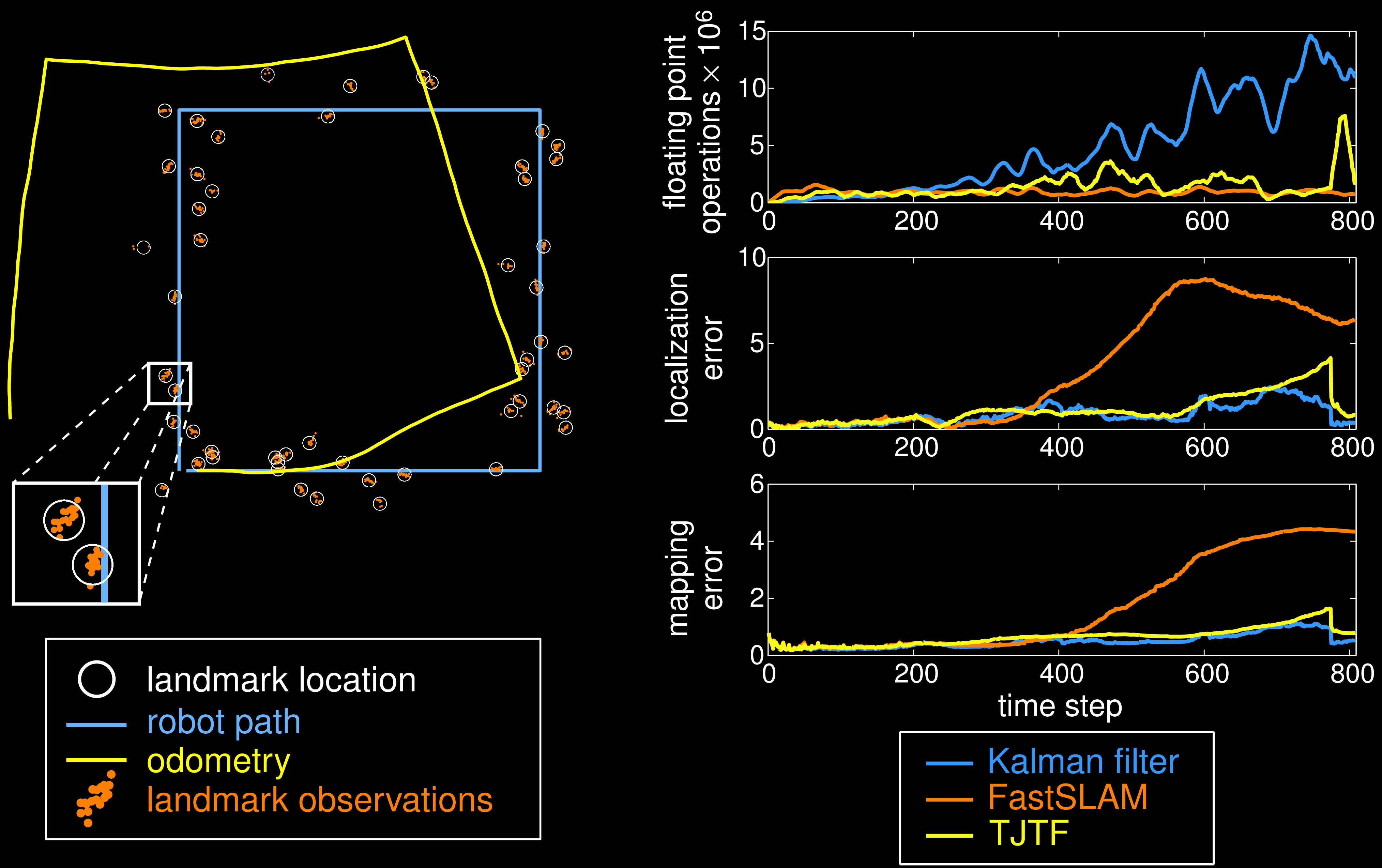
Significance is measured by $D(\phi_S^* \,||\, \phi_S)$, which decreases with distance.



Simulation results



Simulation results



Summary

Thin junction tree filtering:

- a novel algorithm for adaptive approximate filtering in dynamic Bayesian networks
- an elegant solution to the Simultaneous Localization and Mapping problem

More movies and the implementation:

http://www.cs.berkeley.edu/~paskin/slam

Thanks to intel for supporting this research!