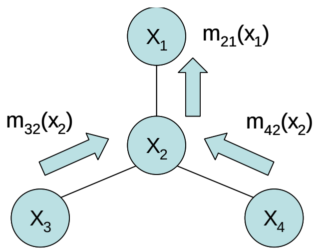
School of Computer Science
Carnegie Mellon

Probabilistic Graphical Models

The Belief Propagation (Sum-Product) Algorithm



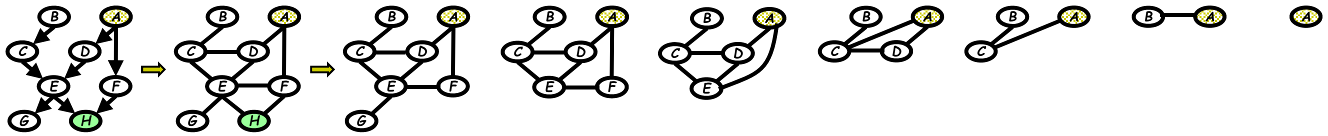
Eric Xing

Lecture 5, January 29, 2014

Reading: KF-chap 10

Pros and Cons of Procedure **Elimination**

- Algebraic elimination ≡ graphical elimination



Complexity

- The overall complexity is determined by the number of the largest elimination clique

  - What is the largest elimination clique? – a pure graph theoretic question
  - **Tree-width** *k*: one less than the smallest achievable value of the cardinality of the largest elimination clique, ranging over all possible elimination ordering
  - “good” elimination orderings lead to **small cliques** and hence reduce complexity (what will happen if we eliminate "e" first in the above graph?)
  - Find the best elimination ordering of a graph --- NP-hard

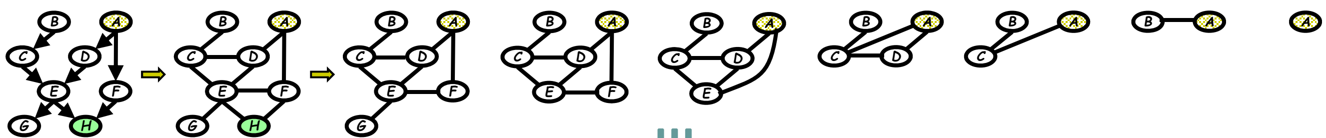
  → Inference is NP-hard

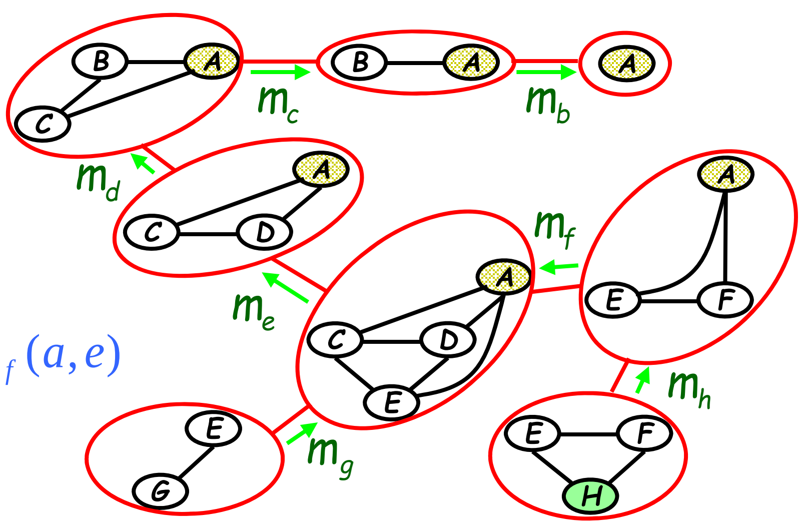
  - But there often exist "obvious" optimal or near-opt elimination ordering

From Elimination to Message Passing

- Our algorithm so far answers only one query (e.g., on one node), do we need to do a complete elimination for every such query?

- Elimination ≡ message passing on a **clique tree**





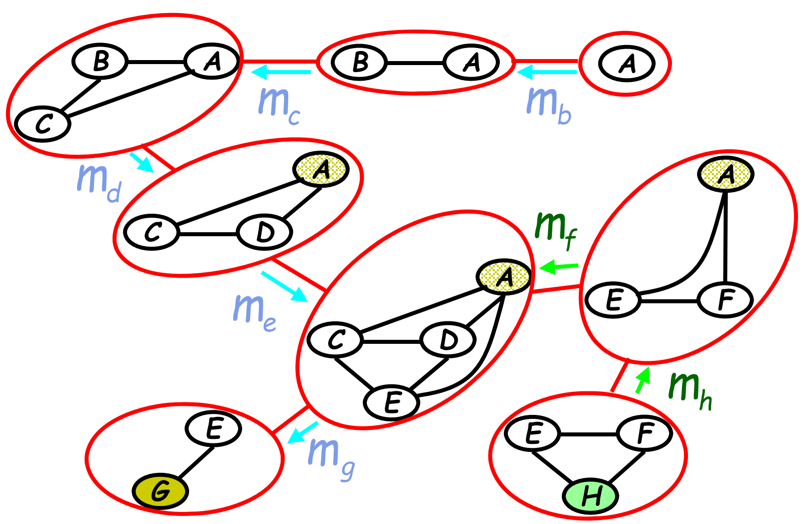
$$m_e(a,c,d) = \sum_e p(e|c,d)m_g(e)m_f(a,e)$$

- Messages can be reused

From Elimination to Message Passing

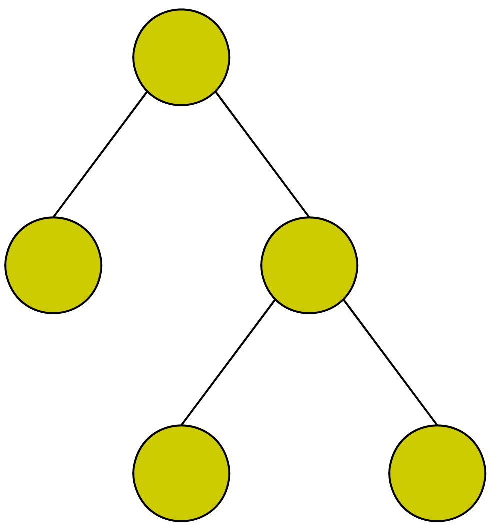
- Our algorithm so far answers only one query (e.g., on one node), do we need to do a complete elimination for every such query?

- Elimination $\equiv$ message passing on a **clique tree**
  - **Another query ...**

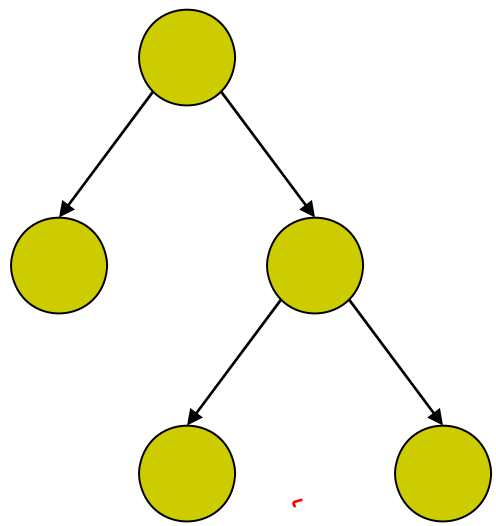


- Messages m_f and m_h are reused, others need to be recomputed

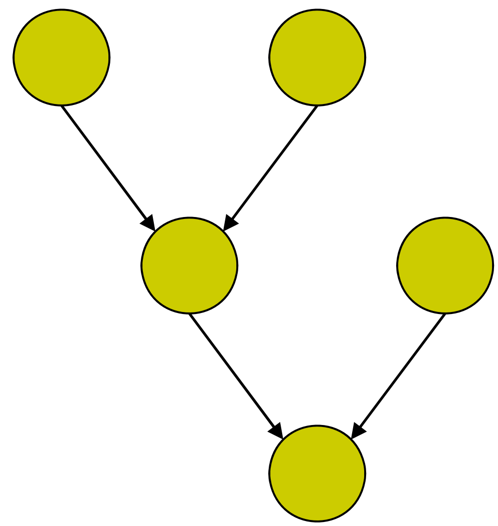
Tree GMs



Undirected tree: a unique path between any pair of nodes



Directed tree: all nodes except the root have exactly one parent



Poly tree: can have multiple parents

We will come back to this later

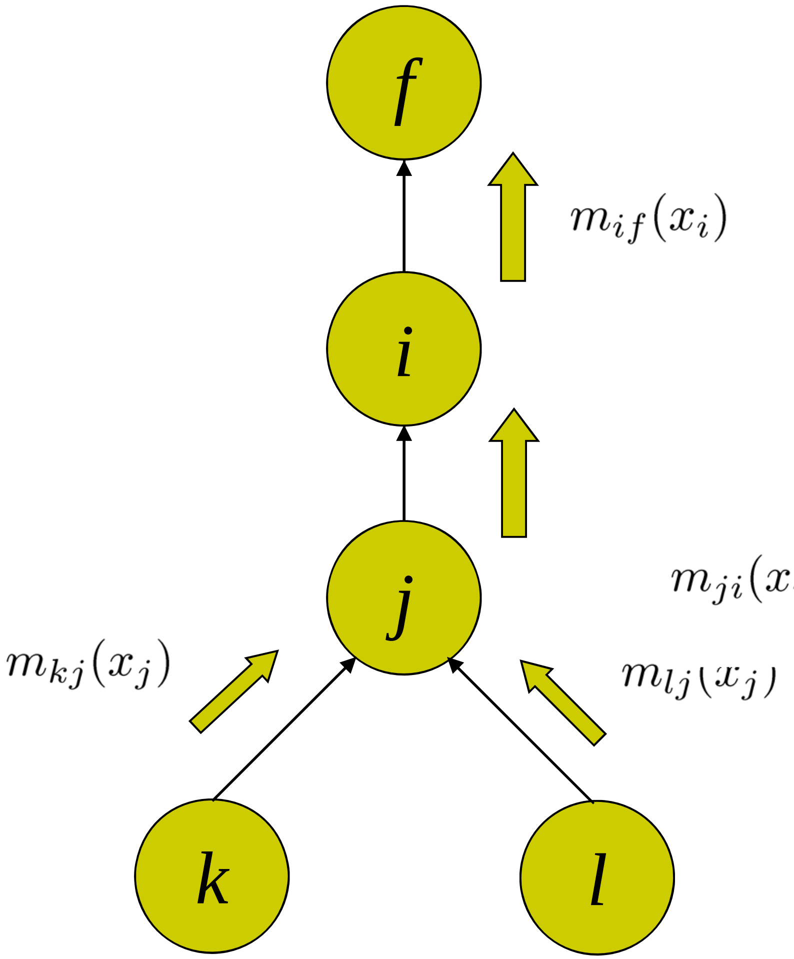
Equivalence of directed and undirected trees

- Any undirected tree can be converted to a directed tree by choosing a root node and directing all edges away from it
- A directed tree and the corresponding undirected tree make the same conditional independence assertions
- Parameterizations are essentially the same.
  - Undirected tree: $$p(x) = \frac{1}{Z}\left(\prod_{i \in V} \psi(x_i) \prod_{(i,j) \in E} \psi(x_i, x_j)\right)$$
  - Directed tree: $$p(x) = p(x_r) \prod_{(i,j) \in E} p(x_j | x_i)$$
  - Equivalence: $$\psi(x_r) = p(x_r); \quad \psi(x_i, x_j) = p(x_j | x_i);$$ $$Z = 1, \quad \psi(x_i) = 1$$
  - Evidence:?

From elimination to message passing

- Recall ELIMINATION algorithm:
  - Choose an ordering $\mathcal{Z}$ in which query node f is the final node
  - Place all potentials on an active list
  - Eliminate node i by removing all potentials containing i, take sum/product over x_i.
  - Place the resultant factor back on the list

Elimination on a tree



Let $m_{ji}(x_i)$ denote the factor resulting from eliminating variables from bellow up to i, which is a function of x_i:

$$m_{ji}(x_i) = \sum_{x_j} \left(\psi(x_j)\psi(x_i, x_j) \prod_{k \in N(j) \backslash i} m_{kj}(x_j) \right)$$

This is reminiscent of a ***message*** sent from j to i.

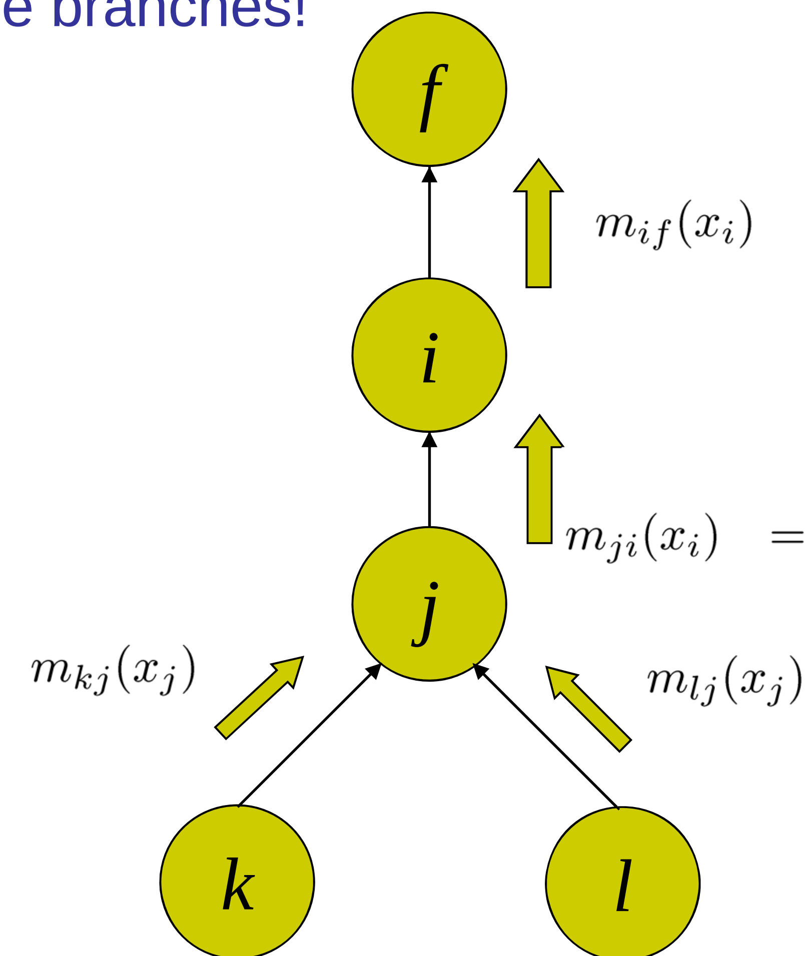
$$m_{ji}(x_i) = \sum_{x_j} \left(\psi(x_j)\psi(x_i, x_j) \prod_{k \in N(j) \backslash i} m_{kj}(x_j) \right)$$

$$p(x_f) \propto \psi(x_f) \prod_{e \in N(f)} m_{ef}(x_f)$$

$m_{ij}(x_i)$ represents a "belief" of x_i from x_j!

Message passing on a tree

- Elimination on trees is equivalent to message passing along tree branches!



$$m_{ji}(x_i) = \sum_{x_j} \left(\psi(x_j)\psi(x_i, x_j) \prod_{k \in N(j) \backslash i} m_{kj}(x_j) \right)$$

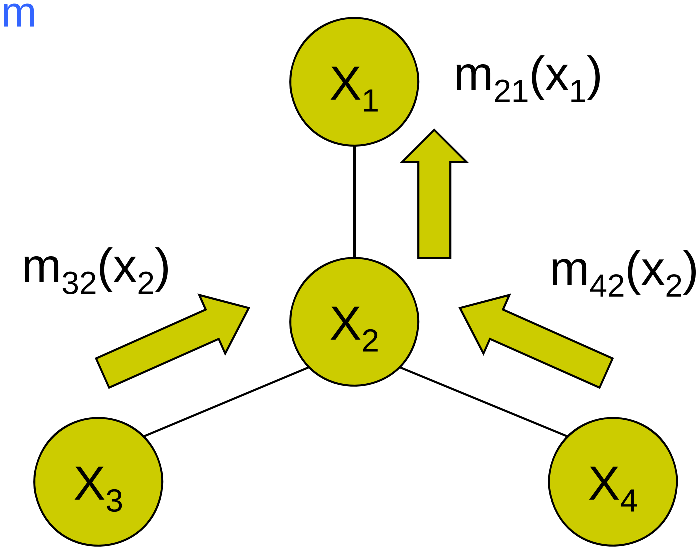
From elimination to message passing

- Recall ELIMINATION algorithm:
  - Choose an ordering $\mathcal{Z}$ in which query node f is the final node
  - Place all potentials on an active list
  - Eliminate node i by removing all potentials containing i, take sum/product over x_i.
  - Place the resultant factor back on the list

- For a TREE graph:
  - Choose query node f as the root of the tree
  - View tree as a directed tree with edges pointing towards leaves from f
  - Elimination ordering based on depth-first traversal
  - Elimination of each node can be considered as message-passing (or Belief Propagation) directly along tree branches, rather than on some transformed graphs
  - → thus, we can use the tree itself as a data-structure to do general inference!!

The message passing protocol:

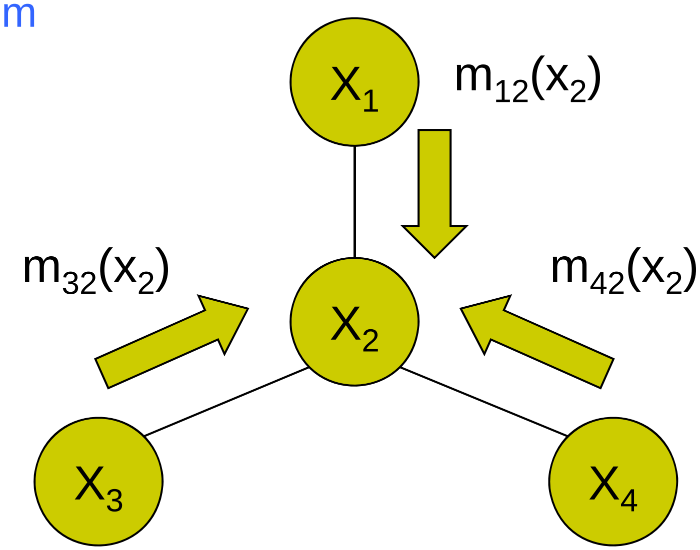
- A node can send a message to its neighbors when (and only when) it has received messages from all its ***other*** neighbors.
- Computing node marginals:
  - Naïve approach: consider each node as the root and execute the message passing algorithm



Computing $P(X_1)$

The message passing protocol:

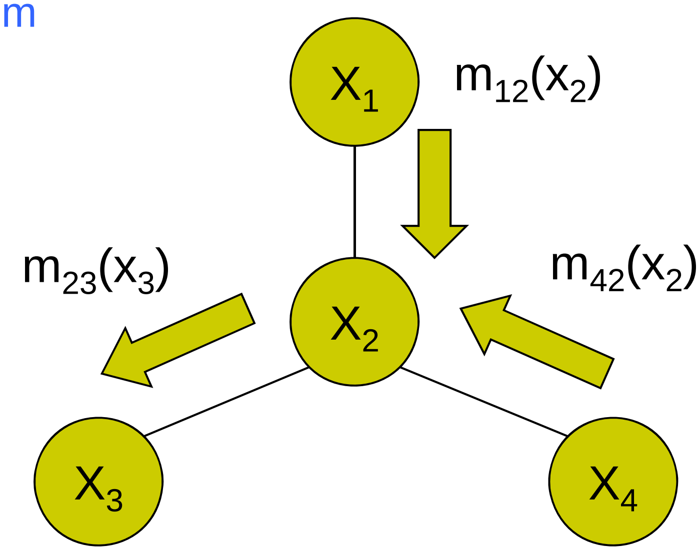
- A node can send a message to its neighbors when (and only when) it has received messages from all its ***other*** neighbors.
- Computing node marginals:
  - Naïve approach: consider each node as the root and execute the message passing algorithm



Computing $P(X_2)$

The message passing protocol:

- A node can send a message to its neighbors when (and only when) it has received messages from all its ***other*** neighbors.
- Computing node marginals:
  - Naïve approach: consider each node as the root and execute the message passing algorithm



Computing $P(X_3)$

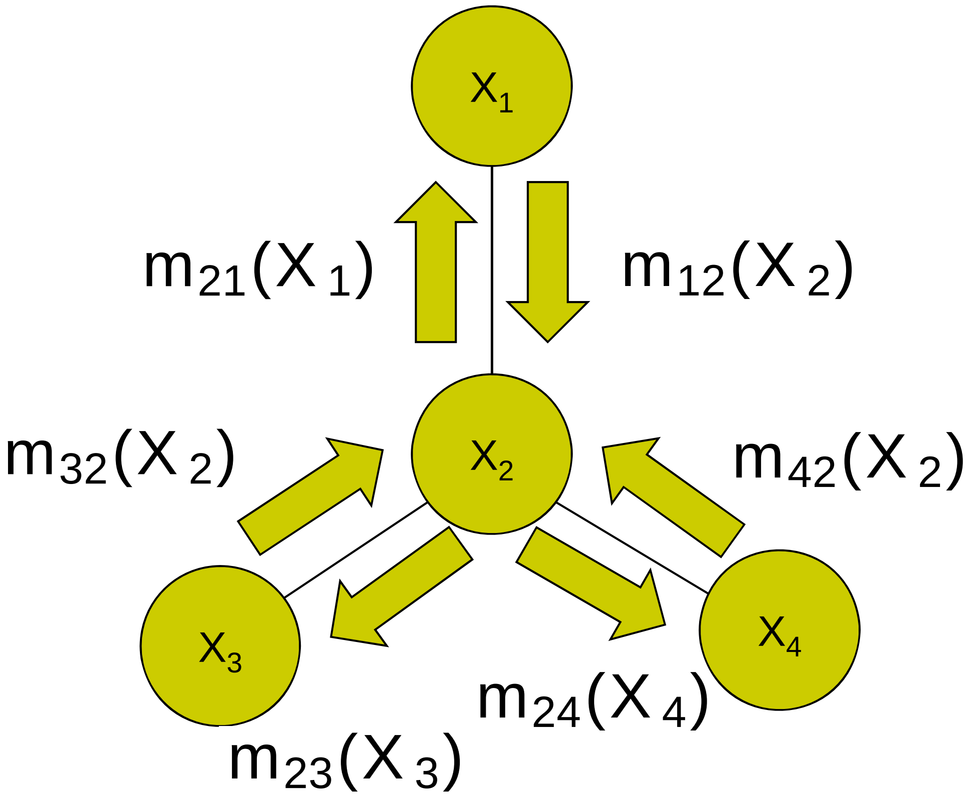
Computing node marginals

- Naïve approach:
  - Complexity: NC
    - N is the number of nodes
    - C is the complexity of a complete message passing

- Alternative dynamic programming approach
  - 2-Pass algorithm (next slide →)
  - Complexity: 2C!

The message passing protocol:

- A two-pass algorithm:



Belief Propagation (SP-algorithm): Sequential implementation

$\textsc{Sum-Product}(\mathcal{T},\ E)$
  $\textsc{Evidence}(E)$
  $f = \textsc{ChooseRoot}(\mathcal{V})$
  for $e \in \mathcal{N}(f)$
    $\textsc{Collect}(f, e)$
  for $e \in \mathcal{N}(f)$
    $\textsc{Distribute}(f, e)$
  for $i \in \mathcal{V}$
    $\textsc{ComputeMarginal}(i)$

$\textsc{Evidence}(E)$
  for $i \in E$
    $\psi^E(x_i) = \psi(x_i)\delta(x_i, \bar{x}_i)$
  for $i \notin E$
    $\psi^E(x_i) = \psi(x_i)$

$\textsc{Collect}(i, j)$
  for $k \in \mathcal{N}(j)\backslash i$
    $\textsc{Collect}(j, k)$
  $\textsc{SendMessage}(j, i)$

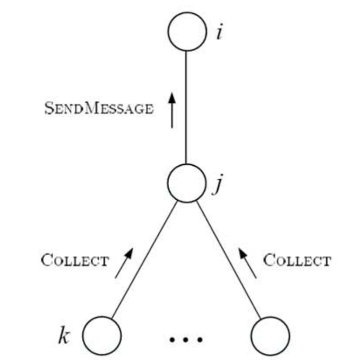
$\textsc{Distribute}(i, j)$
  $\textsc{SendMessage}(i, j)$
  for $k \in \mathcal{N}(j)\backslash i$
    $\textsc{Distribute}(j, k)$

$\textsc{SendMessage}(j, i)$

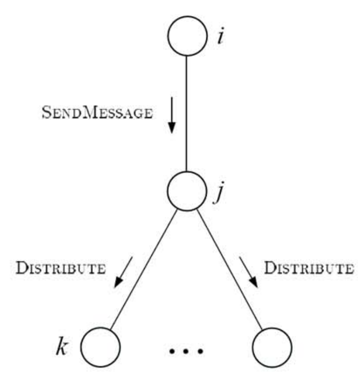
$$m_{ji}(x_i) = \sum_{x_j}(\psi^E(x_j)\psi(x_i, x_j) \prod_{k\in\mathcal{N}(j)\backslash i} m_{kj}(x_j))$$

$\textsc{ComputeMarginal}(i)$

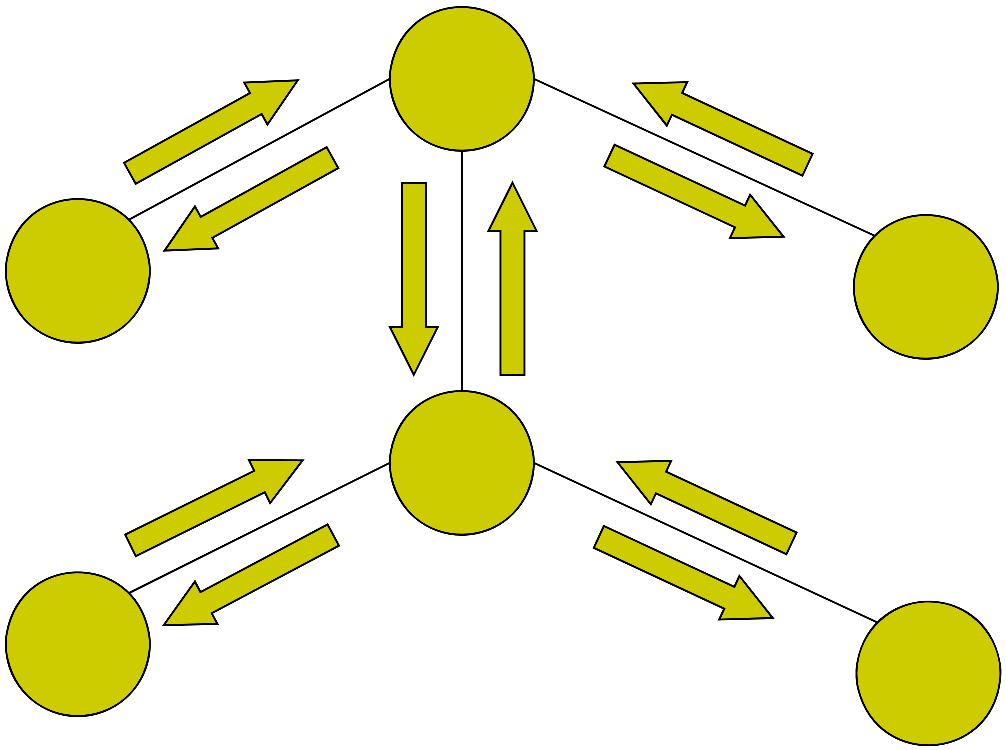
$$p(x_i) \propto \psi^E(x_i) \prod_{j\in\mathcal{N}(i)} m_{ji}(x_i)$$



1



Belief Propagation (SP-algorithm): Parallel synchronous implementation



- For a node of degree d, whenever messages have arrived on any subset of d-1 node, compute the message for the remaining edge and send!
  - A pair of messages have been computed for each edge, one for each direction
  - All incoming messages are eventually computed for each node

Correctness of BP on tree

- Collollary: the synchronous implementation is "non-blocking"

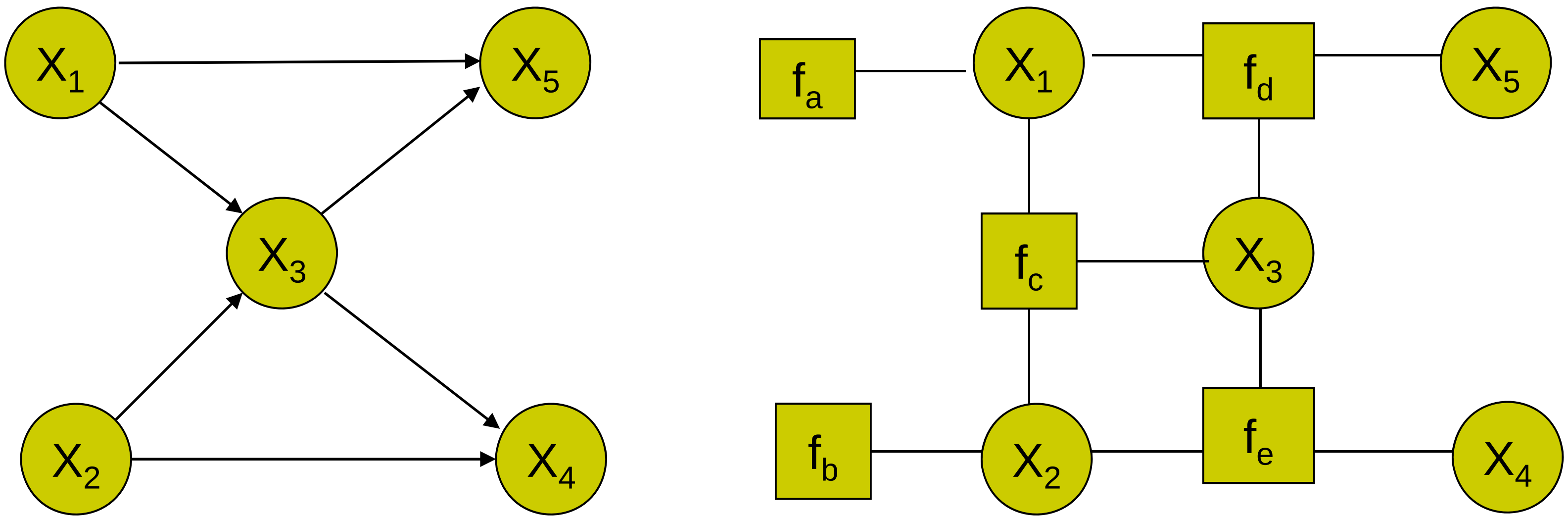
- Thm: The Message Passage Guarantees obtaining all marginals in the tree

$$m_{ji}(x_i) \quad = \quad \sum_{x_j} \left(\psi(x_j)\psi(x_i, x_j) \prod_{k \in N(j) \backslash i} m_{kj}(x_j) \right)$$

- What about non-tree?

Another view of SP: Factor Graph

- Example 1



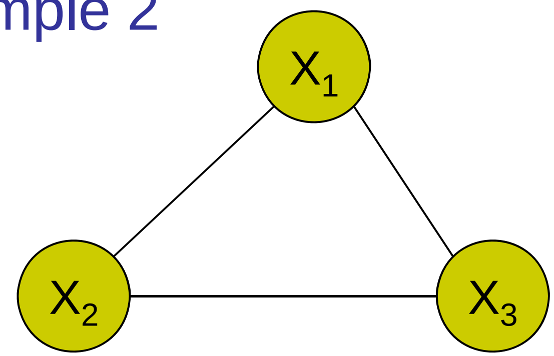
$$P(X_1) \quad P(X_2) \quad P(X_3|X_1,X_2) \quad P(X_5|X_1,X_3) \quad P(X_4|X_2,X_3)$$

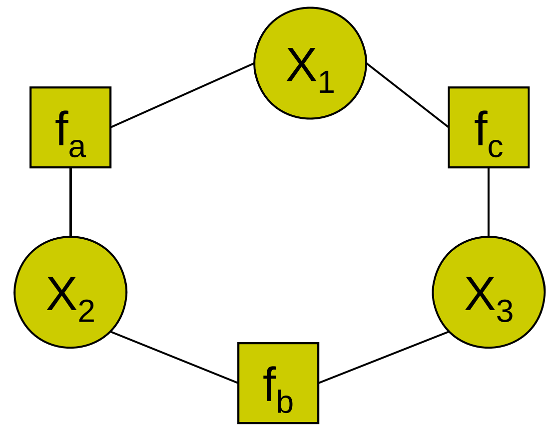
$$\Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow$$

$$f_a(X_1) \quad f_b(X_2) \quad f_c(X_3,X_1,X_2) \quad f_d(X_5,X_1,X_3) \quad f_e(X_4,X_2,X_3)$$

Factor Graphs

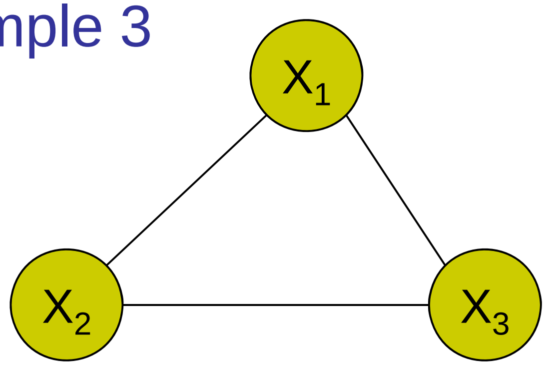
- Example 2

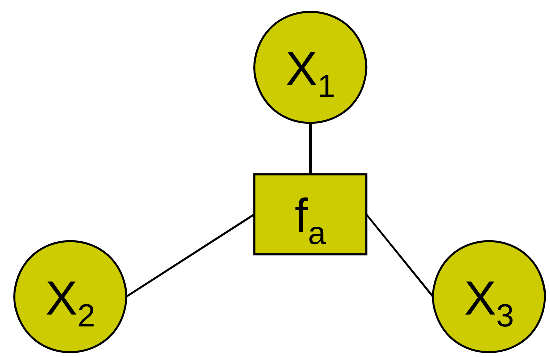




$$\psi(x_1,x_2,x_3) = f_a(x_1,x_2)f_b(x_2,x_3)f_c(x_3,x_1)$$

- Example 3

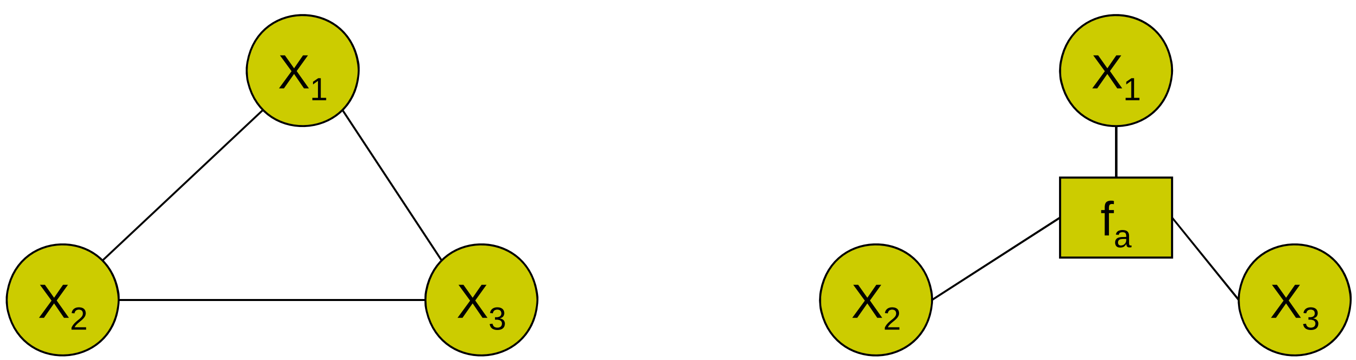




$$\psi(x_1,x_2,x_3) = f_a(x_1,x_2,x_3)$$

Factor Tree

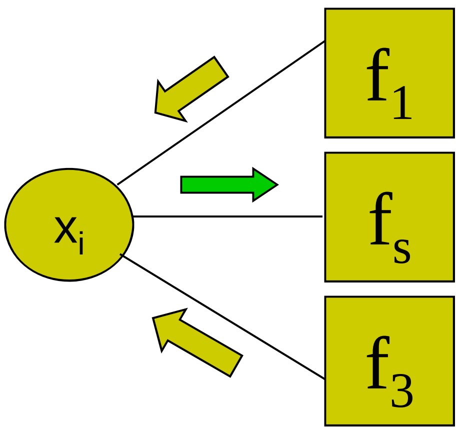
- A Factor graph is a Factor Tree if the undirected graph obtained by ignoring the distinction between variable nodes and factor nodes is an undirected tree

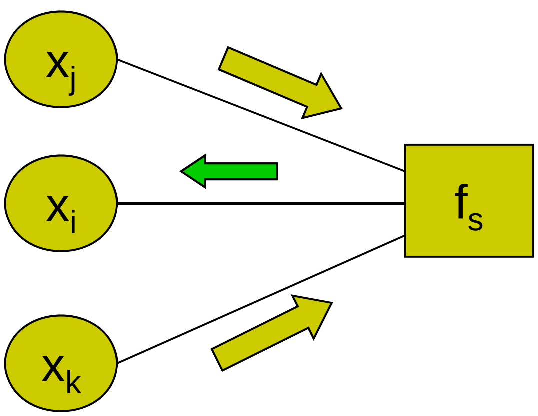


$\psi(x_1,x_2,x_3) = f_a(x_1,x_2,x_3)$

Message Passing on a Factor Tree

- Two kinds of messages
  1. ν: from variables to factors
  2. μ: from factors to variables



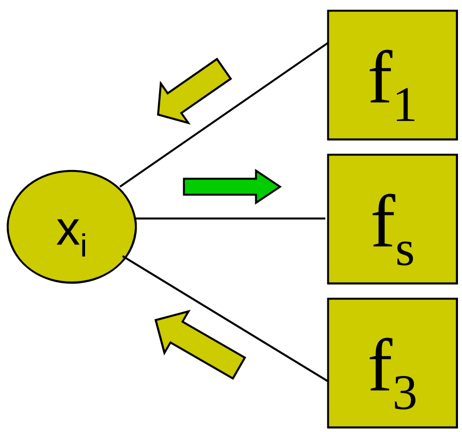


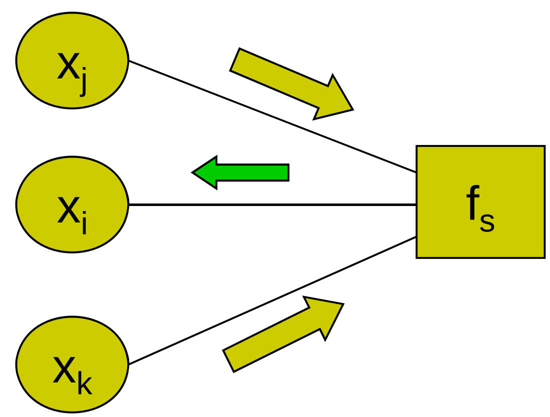
$$\nu_{is}(x_i) = \prod_{t \in \mathcal{N}(i) \backslash s} \mu_{ti}(x_i)$$

$$\mu_{si}(x_i) = \sum_{x_{\mathcal{N}(s)} \backslash i} \left(f_s(x_{\mathcal{N}(s)}) \prod_{j \in \mathcal{N}(s) \backslash i} \nu_{js}(x_j) \right)$$

Message Passing on a Factor Tree, con'd

- Message passing protocol:
  - A node can send a message to a neighboring node only when it has received messages from all its ***other*** neighbors
- Marginal probability of nodes:

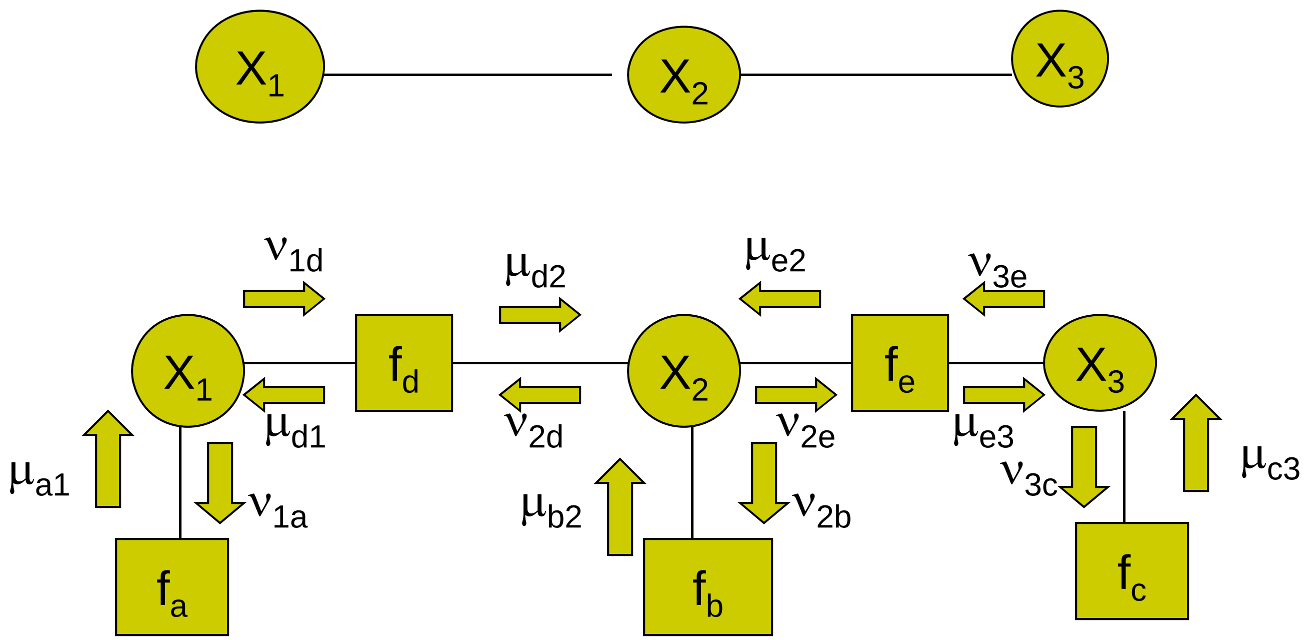




$$P(x_i) \propto \prod_{s \in N(i)} \mu_{si}(x_i)$$

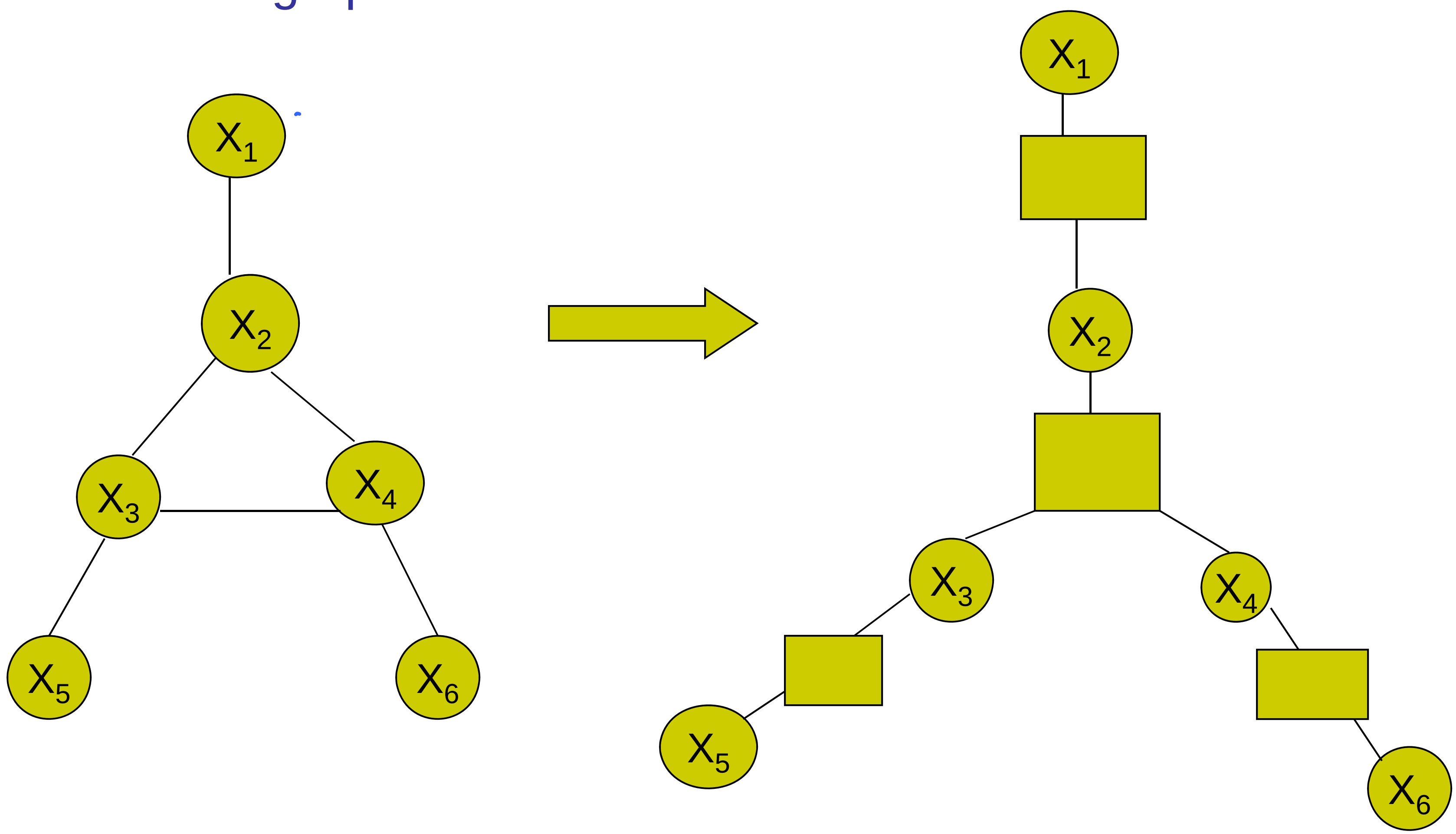
$$\propto \nu_{is}(x_i)\mu_{si}(x_i)$$

BP on a Factor Tree

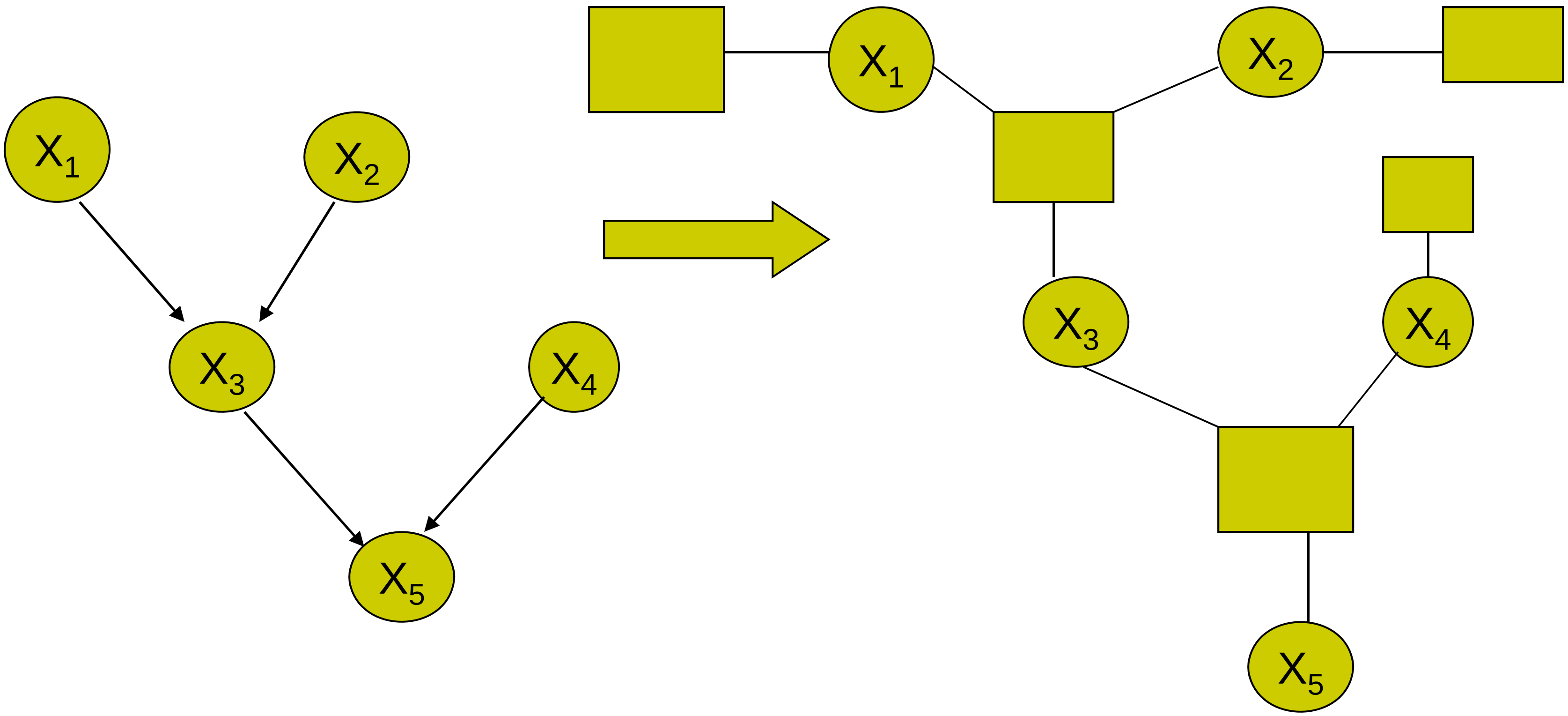


Why factor graph?

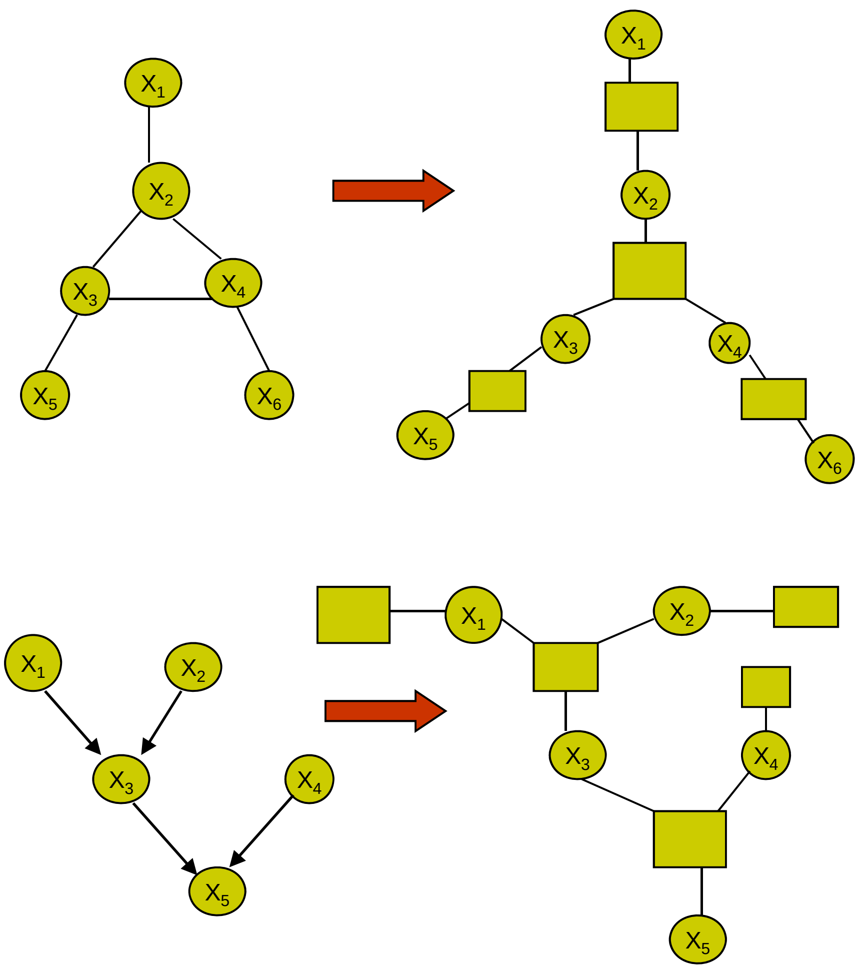
- Tree-like graphs to Factor trees



Poly-trees to Factor trees

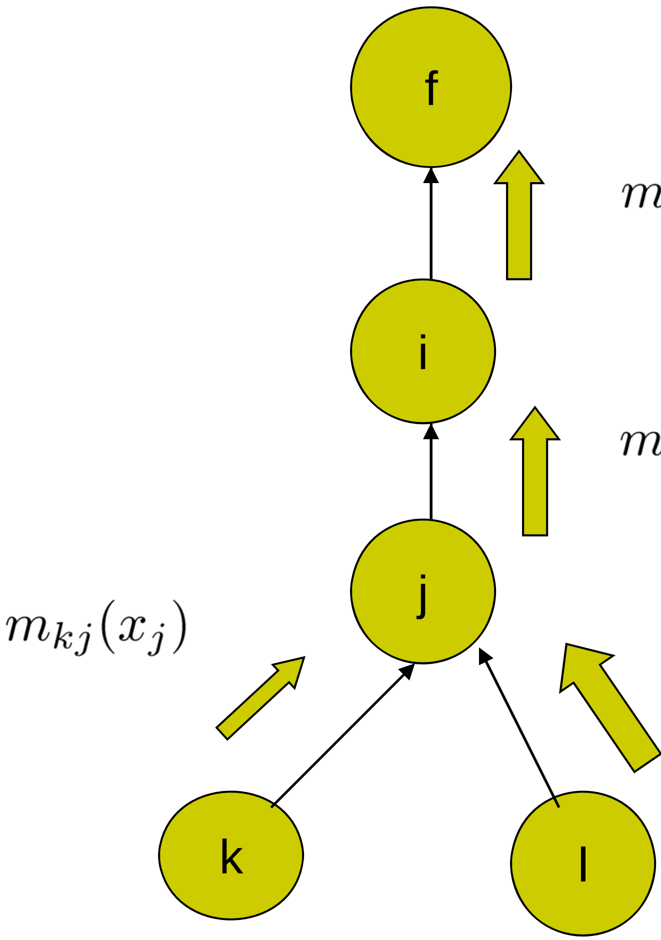


Why factor graph?



- Because FG turns tree-like graphs to factor trees,
- and trees are a data-structure that guarantees correctness of BP !

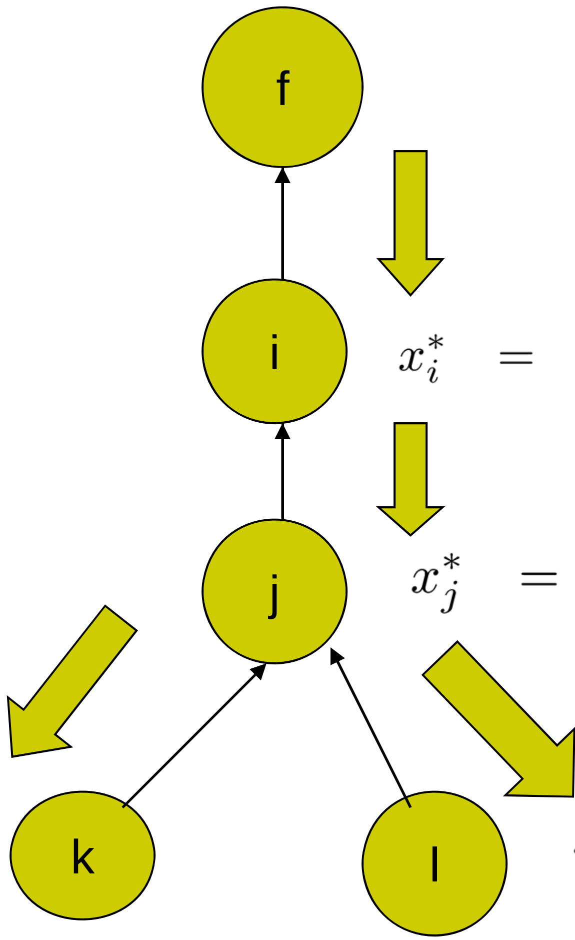
Max-product algorithm: computing MAP probabilities



$$\max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_f}(\psi(x_f) m_{if}(x_f))$$

$$m_{ji}(x_i) = \max_{x_j}\left(\psi(x_j)\psi(x_i, x_j) \prod_{k \in N(j)-i} m_{kj}(x_j)\right)$$

Max-product algorithm:
computing MAP configurations using a final bookkeeping backward pass



$$x_f^* = \arg\max_{x_f}(\psi(x_f)m_{if}(x_f))$$

$$x_i^* = \arg\max_{x_i}\left(\psi(x_i)\psi(x_f^*, x_i)m_{ji}(x_i)\right)$$

$$x_j^* = \arg\max_{x_j}\left(\psi(x_j)\psi(x_i^*, x_j)m_{kj}(x_j)m_{lj}(x_j)\right)$$

$$x_l^* = \arg\max_{x_l}\left(\psi(x_l)\psi(x_l, x_j^*)\right)$$

$$x_k^* = \arg\max_{x_k}\left(\psi(x_k)\psi(x_k, x_j^*)\right)$$

Summary

- Sum-Product algorithm computes singleton marginal probabilities on:
  - Trees
  - Tree-like graphs
  - Poly-trees

- *Maximum a posteriori* configurations can be computed by replacing sum with max in the sum-product algorithm
  - Extra bookkeeping required

Inference on general GM

- Now, what if the GM is not a tree-like graph?

- Can we still directly run

  message-passing protocol along its edges?

- For non-trees, we do not have the guarantee that message-passing will be consistent!

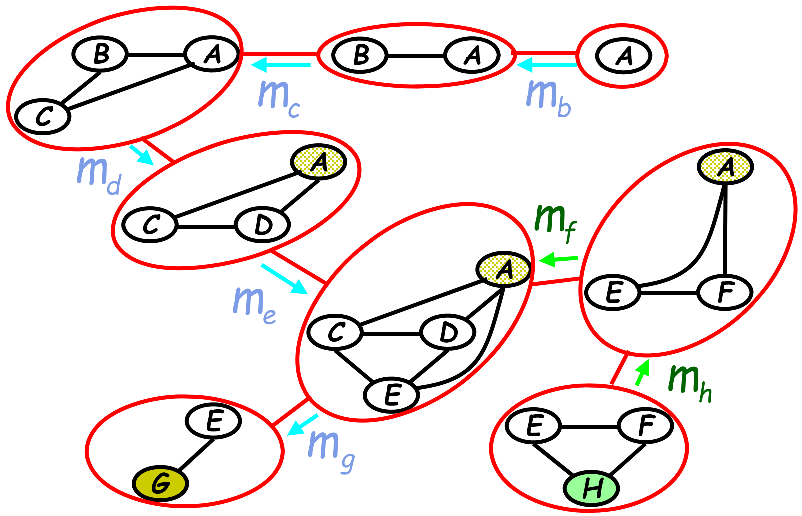
- Then what?
  - Construct a graph data-structure from P that has a tree structure, and run message-passing on it!

→ Junction tree algorithm

Elimination Clique

- Recall that Induced dependency during marginalization is captured in elimination cliques
  - Summation <-> elimination
  - Intermediate term <-> elimination clique

$$
\begin{aligned}
& P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f) \\
\Rightarrow\ & P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)\phi_h(e,f) \\
\Rightarrow\ & P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)\phi_g(e)\phi_h(e,f) \\
\Rightarrow\ & P(a)P(b)P(c|b)P(d|a)P(e|c,d)\phi_f(a,e) \\
\Rightarrow\ & P(a)P(b)P(c|b)P(d|a)\phi_e(a,c,d) \\
\Rightarrow\ & P(a)P(b)P(c|b)\phi_d(a,c) \\
\Rightarrow\ & P(a)P(b)\phi_c(a,b) \\
\Rightarrow\ & P(a)\phi_b(a) \\
\Rightarrow\ & \phi(a)
\end{aligned}
$$



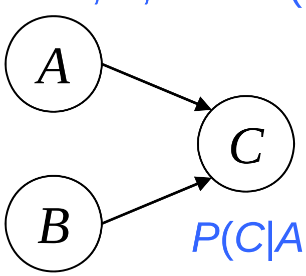
  - Can this lead to an generic inference algorithm?

Moral Graph

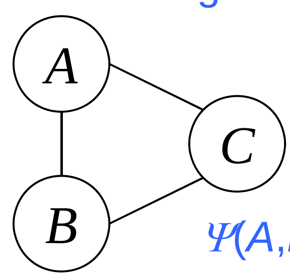
- Note that for both directed GMs and undirected GMs, the joint probability is in a product form:

BN: $P(\mathbf{X}) = \prod_{i=1:d} P(X_i \mid \mathbf{X}_{\pi_i})$ **MRF:** $P(\mathbf{X}) = \frac{1}{Z} \prod_{c \in C} \psi_c(\mathbf{X}_c)$

- So let's convert local conditional probabilities into potentials; then the second expression will be generic, but how does this operation affect the directed graph?
  - We can think of a conditional probability, e.g,. $P(C|A,B)$ as a function of the three variables A, B, and C (we get a real number of each configuration):



$P(C|A,B)$

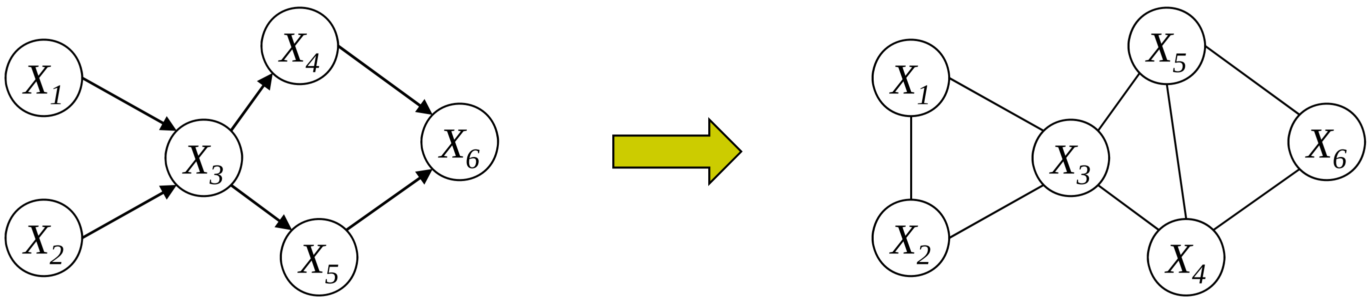


$\Psi(A,B,C) = P(C|A,B)$

  - Problem: But a node and its parent are not generally in the same clique in a BN
  - Solution: Marry the parents to obtain the "moral graph"

Moral Graph (cont.)

- Define the potential on a clique as the product over all conditional probabilities contained within the clique
- Now the product of potentials gives the right answer:



$$
\begin{aligned}
& P(X_1, X_2, X_3, X_4, X_5, X_6) \\
&= P(X_1)P(X_2)P(X_3 \mid X_1, X_2)P(X_4 \mid X_3)P(X_5 \mid X_3)P(X_6 \mid X_4, X_5) \\
&= \psi(X_1, X_2, X_3)\psi(X_3, X_4, X_5)\psi(X_4, X_5, X_6)
\end{aligned}
$$

where

$$\psi(X_1, X_2, X_3) = P(X_1)P(X_2)P(X_3 \mid X_1, X_2)$$

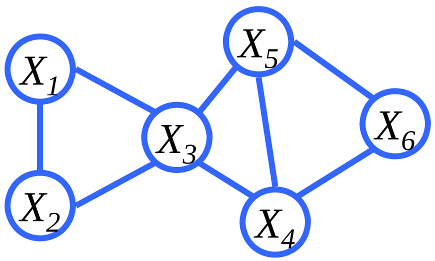
$$\psi(X_3, X_4, X_5) = P(X_4 \mid X_3)P(X_5 \mid X_3)$$

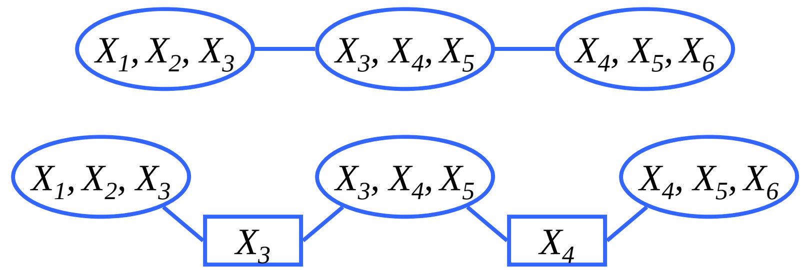
$$\psi(X_4, X_5, X_6) = P(X_6 \mid X_4, X_5)$$

Note that here the interpretation of potential is ambivalent:
it can be either *marginals* or *conditionals*

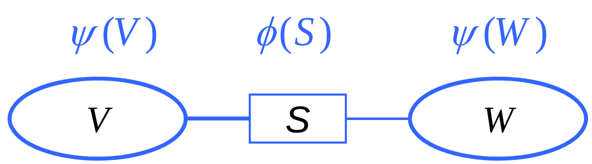
Clique trees

- A clique tree is an (undirected) tree of cliques





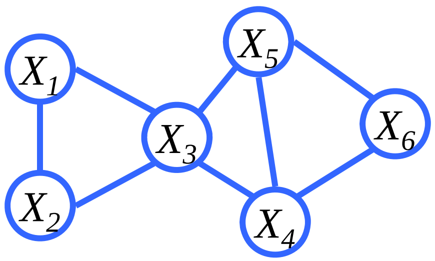
- Consider cases in which two neighboring cliques V and W have an overlap S (e.g., (X_1, X_2, X_3) overlaps with (X_3, X_4, X_5)),

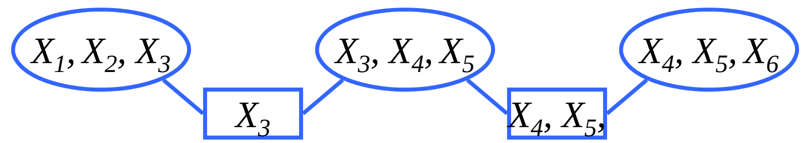


- Now we have an alternative representation of the joint in terms of the potentials:

Clique trees

- A clique tree is an (undirected) tree of cliques





- The alternative representation of the joint in terms of the potentials:

$$
\begin{aligned}
& P(X_1, X_2, X_3, X_4, X_5, X_6) \\
&= P(X_1)P(X_2)P(X_3 \mid X_1, X_2)P(X_4 \mid X_3)P(X_5 \mid X_3)P(X_6 \mid X_4, X_5) \\
&= P(X_1, X_2, X_3)\frac{P(X_3, X_4, X_5)}{P(X_3)}\frac{P(X_4, X_5, X_6)}{P(X_4, X_5)} \\
&= \psi(X_1, X_2, X_3)\frac{\psi(X_3, X_4, X_5)}{\phi(X_3)}\frac{\psi(X_4, X_5, X_6)}{\phi(X_4, X_5)}
\end{aligned}
$$

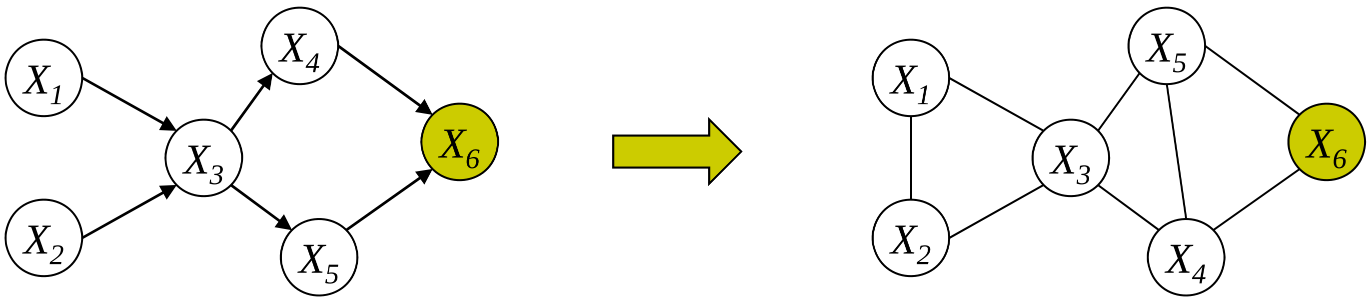
Now each potential is isomorphic to the *cluster marginal* of the attendant set of variables

- Generally:

$$\boxed{P(\mathbf{X}) = \frac{\prod_C \psi_C(\mathbf{X}_C)}{\prod_S \phi_S(\mathbf{X}_S)}}$$

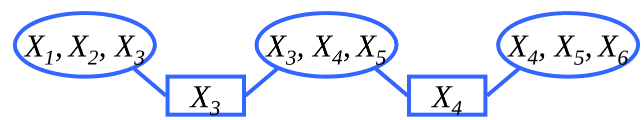
Why this is useful?

- Propagation of probabilities
  - Now suppose that some evidence has been "absorbed" (i.e., certain values of some nodes have been observed). How do we propagate this effect to the rest of the graph?



  - What do we mean by propagate?

    Can we adjust all the potentials $\{\psi\}$, $\{\phi\}$ so that they still represent the correct cluster marginals (or unnormalized equivalents) of their respective attendant variables?



  - Utility?

$$P(X_1 \mid X_6 = x_6) = \sum_{X_2, X_3} \psi(X_1, X_2, X_3)$$

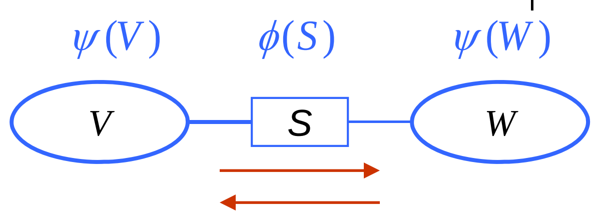
$$P(X_3 \mid X_6 = x_6) = \phi(X_3)$$

$$P(x_6) = \sum_{X_4, X_5} \psi(X_4, X_5, x_6)$$

Local operations!

Local Consistency

- We have two ways of obtaining $p(S)$



$$P(S) = \sum_{V \setminus S} \psi(V) \qquad P(S) = \sum_{W \setminus S} \psi(W)$$

and they must be the same

- The following update-rule ensures this:
  - Forward update: $\phi_S^* = \sum_{V \setminus S} \psi^*{}_V \qquad \psi_W^* = \frac{\phi_S^*}{\phi_S} \psi_W$
  - Backward update $\phi_S^{**} = \sum_{W \setminus S} \psi_W^* \qquad \psi_V^{**} = \frac{\phi_S^{**}}{\phi_S^*} \psi_V^*$

- Two important identities can be proven

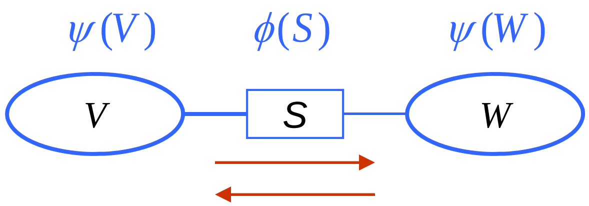
$$\sum_{V \setminus S} \psi_V^{**} = \sum_{W \setminus S} \psi_W^* = \phi_S^{**}$$

Local Consistency

$$\frac{\psi_V^* \psi_W^*}{\phi_S^*} = \frac{\psi_V^{**} \psi_W^{**}}{\phi_S^{**}} = \frac{\psi_V \psi_W}{\phi_S}$$

Invariant Joint

Message Passing Algorithm



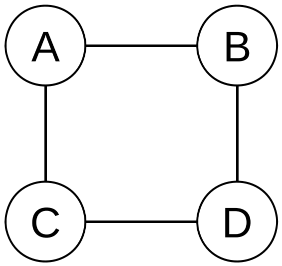
$$\phi_S^{*} = \sum_{V \setminus S} \psi_V^{*} \qquad \psi_W^{*} = \frac{\phi_S^{*}}{\phi_S} \psi_W$$

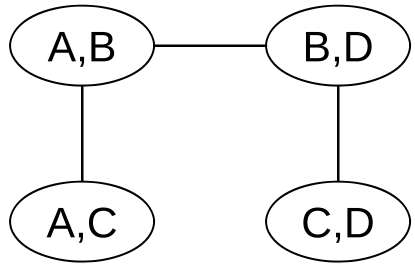
$$\phi_S^{**} = \sum_{W \setminus S} \psi_W^{*} \qquad \psi_V^{**} = \frac{\phi_S^{**}}{\phi_S^{*}} \psi_V^{*}$$

- This simple local message-passing algorithm on a clique tree defines the general probability propagation algorithm for directed graphs!
  - Many interesting algorithms are special cases:
    - Forward-backward algorithm for hidden Markov models,
    - Kalman filter updates
    - Pealing algorithms for probabilistic trees
  - The algorithm seems reasonable. Is it correct?

A problem

- Consider the following graph and a corresponding clique tree

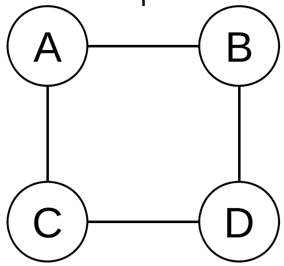




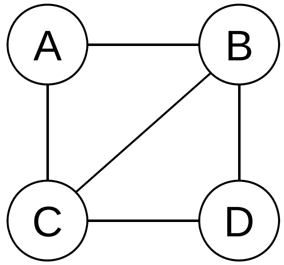
  - Note that C appears in two non-neighboring cliques
- *Question*: with the previous message passage, can we ensure that the probability associated with C in these two (non-neighboring) cliques consistent?
- Answer: No. It is not true that in general local consistency implies global consistency
- What else do we need to get such a guarantee?

Triangulation

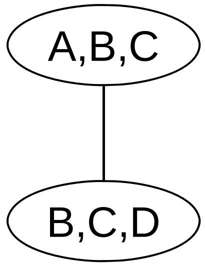
- A triangulated graph is one in which *no cycles* with four or more nodes exist in which there is no *chord*



- We triangulate a graph by adding chords:

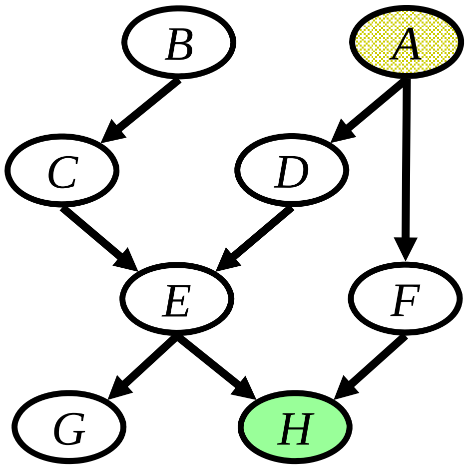


- Now we no longer have our global inconsistency problem.
  - A clique tree for a triangulated graph has the *running intersection property*: If a node appears in two cliques, it appears everywhere on the path between the cliques
  - Thus local consistency implies global consistency



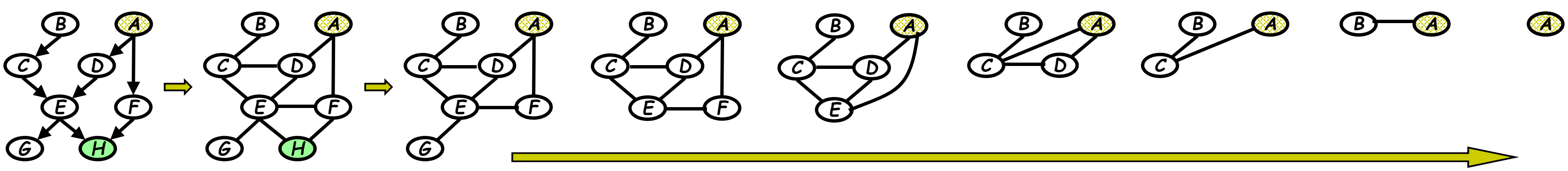
Junction trees

- A clique tree for a triangulated graph is referred to as a *junction tree*
- In junction trees, local consistency implies global consistency. Thus the local message-passing algorithms is (provably) correct
- It is also possible to show that *only* triangulated graphs have the property that their clique trees are junction trees. Thus if we want local algorithms, we *must* triangulate
- Are we now all set?
  - How to triangulate?
  - The complexity of building a JT depends on how we triangulate!!
  - Consider this network: it turns out that we will need to pay an O(2^4) or O(2^6) cost depending on how we triangulate!



How to triangulate

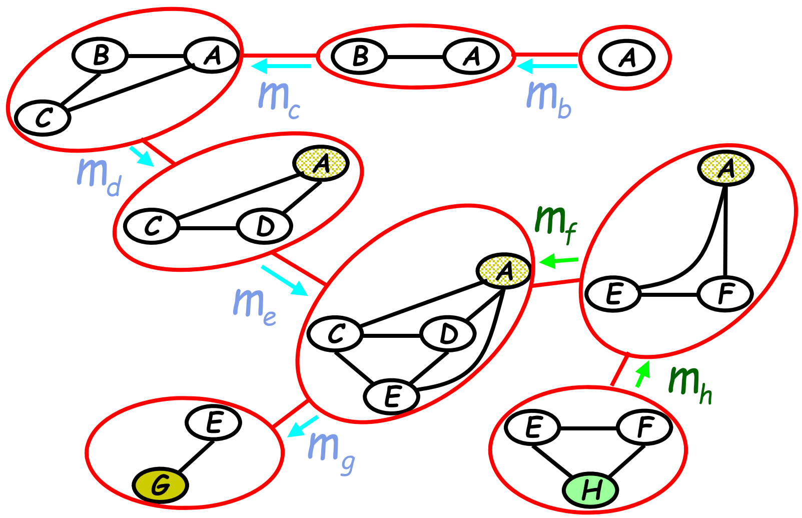
- A graph elimination algorithm



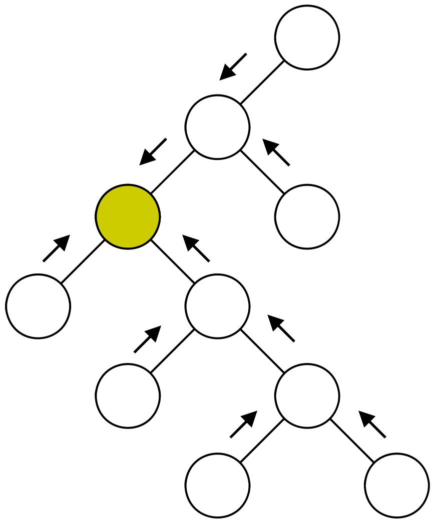
moralization graph elimination

- Intermediate terms correspond to the cliques resulted from elimination
  - “good” elimination orderings lead to **small cliques** and hence reduce complexity (what will happen if we eliminate "e" first in the above graph?)
  - finding the optimum ordering is NP-hard, but for many graph optimum or near-optimum can often be heuristically found

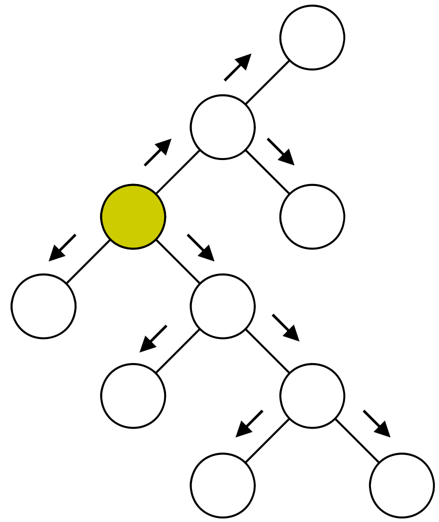
A junction tree



Message-passing algorithms



collect



distribute

- Message update
  - The Hugin update
  - The Shafer-Shenoy update

$$\phi_S^* = \sum_{V \setminus S} \psi_V \qquad \psi_W^* = \frac{\phi_S^*}{\phi_S} \psi_W$$

$$m_{i \to j}(S_{ij}) = \sum_{C_i \setminus S_{ij}} \psi_{C_i} \prod_{k \neq j} m_{k \to i}(S_{ki})$$

A Sketch of the Junction Tree Algorithm

- **The algorithm**

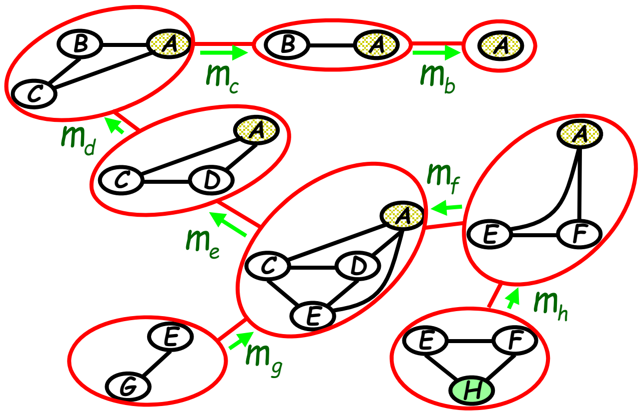
  1. Moralize the graph (trivial)
  2. Triangulate the graph (good heuristic exist, but actually NP hard)
  3. Build a clique tree (e.g., using a maximum spanning tree algorithm
  4. Propagation of probabilities --- a local message-passing protocol

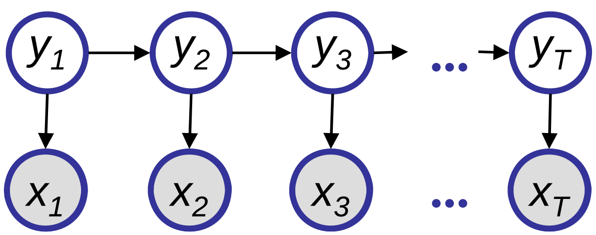
- Results in marginal probabilities of all cliques --- solves all queries in a single run
- A **generic** exact inference algorithm for any GM
- **Complexity**: exponential in the size of the maximal clique --- a good elimination order often leads to small maximal clique, and hence a good (i.e., thin) JT

Recall the Elimination and Message Passing Algorithm

- Elimination $\equiv$ message passing on a **clique tree**

$$m_e(a,c,d) = \sum_e p(e|c,d)m_g(e)m_f(a,e)$$



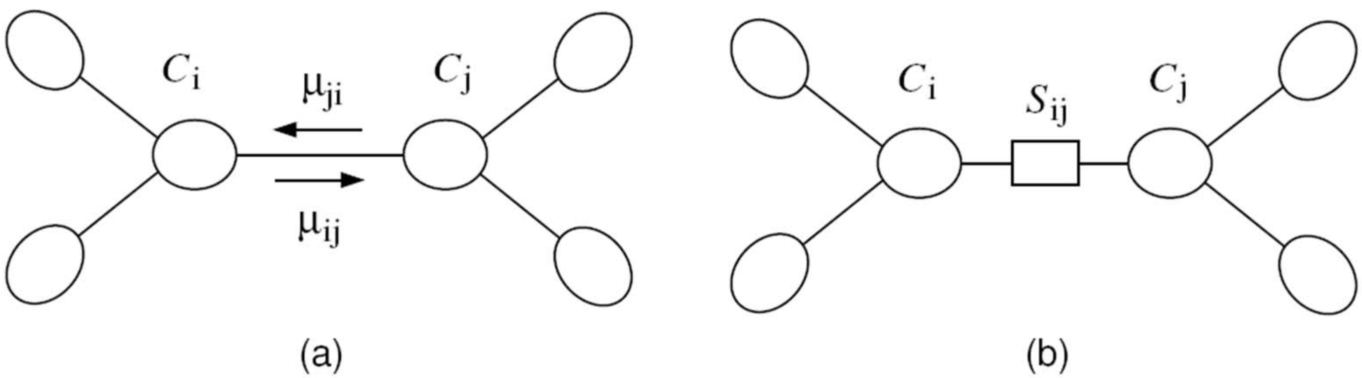


$$P(\mathbf{x}) = \sum_k \alpha_T^k$$

$$\alpha_t^k = p(x_t \mid y_t^k = 1)\sum_i \alpha_{t-1}^i a_{i,k}$$

Shafer Shenoy for HMMs

- Recap: Shafer-Shenoy algorithm



  - Message from clique *i* to clique *j* :

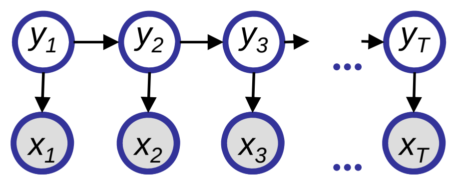
$$\mu_{i\to j} = \sum_{C_i \setminus S_{ij}} \psi_{C_i} \prod_{k\neq j} \mu_{k\to i}(S_{ki})$$

  - Clique marginal

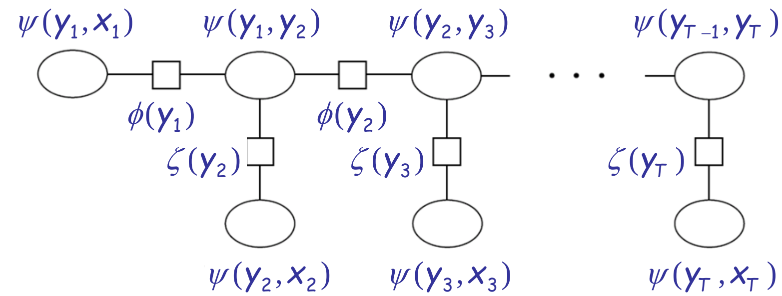
$$p(C_i) \propto \psi_{C_i} \prod_{k} \mu_{k\to i}(S_{ki})$$

Message Passing for HMMs (cont.)

- A junction tree for the HMM



$\Rightarrow$

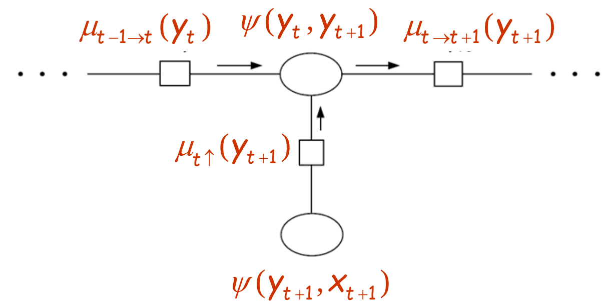


- Rightward pass

$$\mu_{t\to t+1}(y_{t+1}) = \sum_{y_t} \psi(y_t, y_{t+1})\mu_{t-1\to t}(y_t)\mu_{t\uparrow}(y_{t+1})$$

$$= \sum_{y_t} p(y_{t+1} \mid y_t)\mu_{t-1\to t}(y_t)p(x_{t+1} \mid y_{t+1})$$

$$= p(x_{t+1} \mid y_{t+1})\sum_{y_t} a_{y_t, y_{t+1}}\mu_{t-1\to t}(y_t)$$

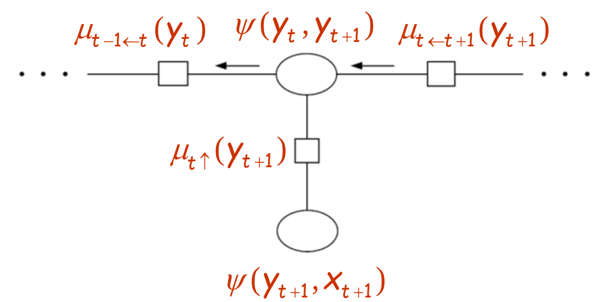


  - This is exactly the *forward algorithm*!

- Leftward pass …

$$\mu_{t-1\leftarrow t}(y_t) = \sum_{y_{t+1}} \psi(y_t, y_{t+1})\mu_{t\leftarrow t+1}(y_{t+1})\mu_{t\uparrow}(y_{t+1})$$

$$= \sum_{y_{t+1}} p(y_{t+1} \mid y_t)\mu_{t\leftarrow t+1}(y_{t+1})p(x_{t+1} \mid y_{t+1})$$



  - This is exactly the *backward algorithm*!

Summary

- Junction tree data-structure for exact inference on general graphs
- Two methods
  - Shafer-Shenoy
  - Belief-update or Lauritzen-Speigelhalter
- Constructing Junction tree from chordal graphs
  - Maximum spanning tree approach