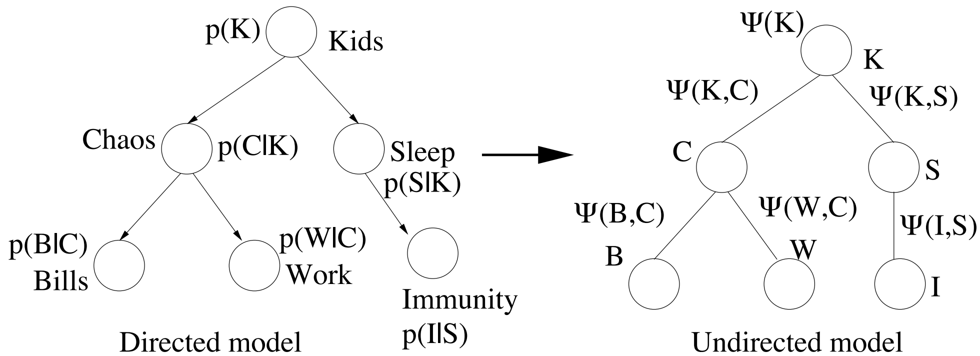
Lecture 7: More on variable elimination

- The special case of trees: message passing
- Variable elimination as a graph operation
- Clique trees
- Junction trees

Recall from last time

- Inference is the process of computing conditional probabilities for query variables given evidence
- Variable elimination is an exact inference procedure based on two ideas:
  - Re-arranging the sums and products that need to be computed
  - Caching the result of intermediate computations
- The complexity is order $n \cdot 2^k$ where n is the number of variables in the network and k is the largest number of variables present in a factor
- The worst-case is having large v-structures, where eliminating the bottom node creates large factors.

An interesting special case: Directed trees



- Directed trees are such that their moral graph is a tree
- We can parameterize the corresponding undirected model by:

$$\Psi(\text{root}) = p(\text{root}) \text{ and } \Psi(x_j, x_i) = p(x_j|x_i)$$

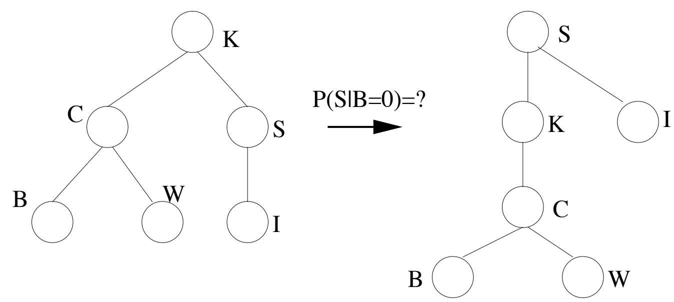
for any nodes such that X_i is the parent of X_j

From undirected to directed trees

- Any undirected tree can be converted into a directed one by picking a root and directing arcs from there outwards
- We will parameterize an undirected tree by $\Psi(x_i)$, for all nodes i, and $\Psi(x_i, x_j)$, for all arcs (X_i, X_j)
- If we want to compute $p(Y|E)$, we introduce the evidence potential $\delta(x_i, \hat{x_i})$, for all evidence variables $X_i \in E$
- The potentials now become:

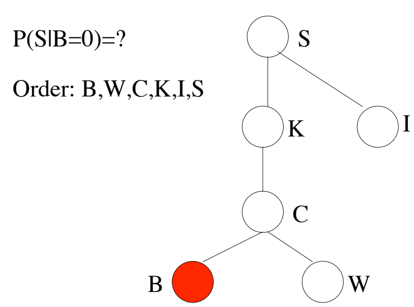
$$\psi^E(x_i) = \begin{cases} \psi(x_i)\delta(x_i, \hat{x_i}) & \text{if } X_i \in E \\ \psi(x_i), & \text{otherwise} \end{cases}$$

Variable elimination on undirected trees



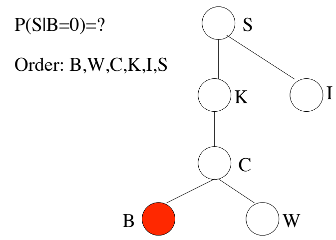
- The query node becomes root
- Traverse the resulting tree *depth-first*
- A node can only be eliminated *after all its children* have been eliminated
- What orderings arise in our example?

Intermediate factors



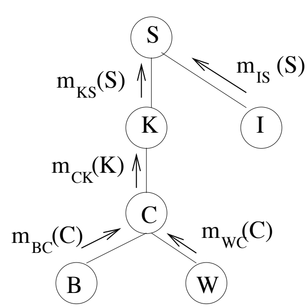
- Consider nodes C and K, which are connected
- C will be eliminated before K
- When we eliminate C, and create factor m_C, what potentials will get out of the active list? What variables will m_C depend on?

Intermediate factors



- $\psi(K, C)$ and $\psi^E(C)$ will have to be eliminated
- None of the factors that will be eliminated can reference B or W, since they would have been eliminated already
- None of the factors can reference I or S, (variables outside C's subtree), because of tree-ness
- So the factor that we create will be a *function of K only*!
- We can view this as a *message* computed by C and passed on to K. Call it $m_{CK}(K)$.

Message passing



The message passed by C to K will be:

$$m_{CK}(K) = \sum_c \left(\psi^E(c)\psi(c, k)m_{BC}(c)m_{WC}(c) \right)$$

where $m_{BC}(C)$ and $m_{WC}(C)$ are the messages from B and C.

Variable elimination for trees

- To eliminate node X_j, we have:

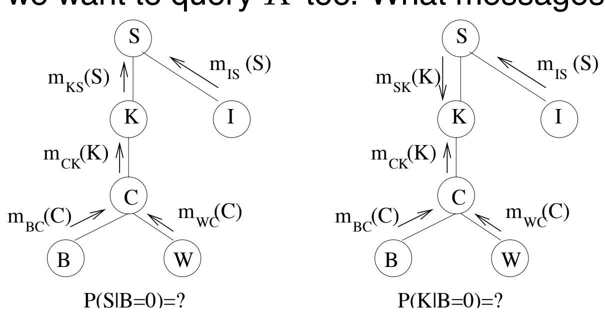
$$m_{ji}(x_i) = \sum_{x_j} \left(\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \text{neighbors}(x_j)-\{x_i\}} m_{kj}(x_j) \right)$$

- The desired probability is computed as:

$$p(y|\hat{x}_E) \propto \psi^E(y) \prod_{k \in \text{neighbors}(Y)} m_{ky}(y)$$

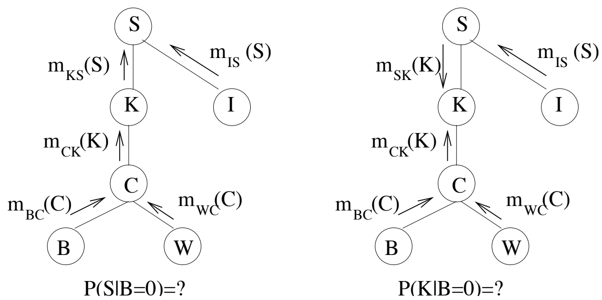
What if we want to query more variables?

- Suppose we want to query K too. What messages are needed?



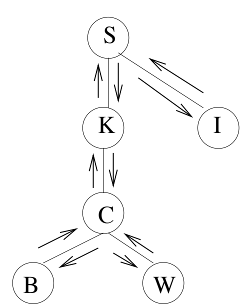
What if we want to query more variables?

- Suppose we want to query K too. What messages are needed?



- Note that almost all messages are the same!
- Key idea: *messages can be re-used* for the computation of other queries

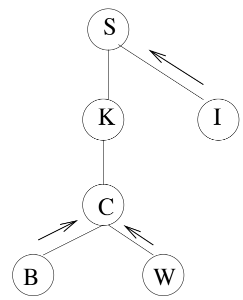
Computing all probabilities



- Because messages can be re-used, we can compute *all* conditional probabilities by computing all messages!
- Note that the number of messages is not too big
- We can use our previous equations to compute messages, but we need a protocol for when to compute them

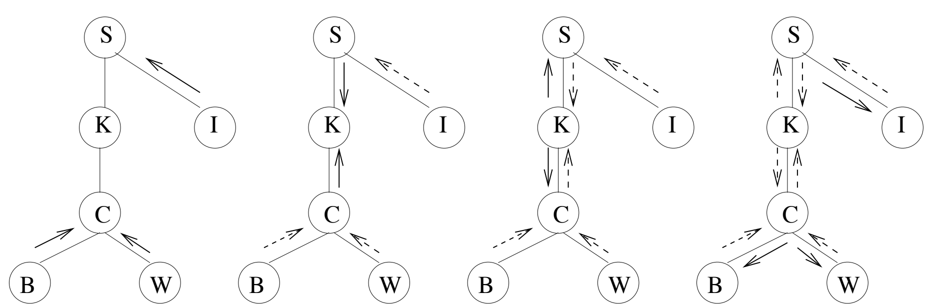
Message-passing protocol

- A node can send a message to a neighbor *after it has received* the messages from *all its other neighbors*.
- Synchronous parallel implementation: any node with d neighbors sends a message after receiving messages on $d-1$ edges



- What messages are sent next?

Message passing example

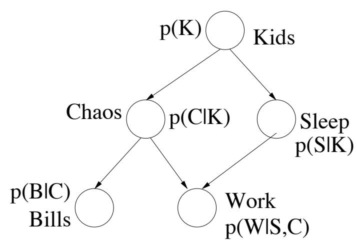


- Past messages are dashed, current messages are in solid arrow.
- This is called the **sum-product** algorithm for trees

Sequential implementation of the sum-product algorithm

1. Introduce the evidence (by putting in the evidence potentials)
2. Choose any node as root
3. Inward pass: Send all messages toward the root
4. Outward pass: Send all messages outward from the root
5. Compute the probabilities at all the nodes

Example: Variable elimination in a Bayes net



Compute $p(B|S=0)$?

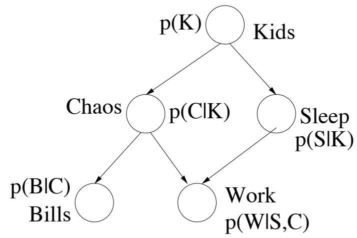
1. Fix a variable ordering, e.g. K, C, S, W, B
2. Initialize the active factors list:

$$p(K), p(C|K), p(S|K), p(W|S,C), p(B|C), \delta(S,0)$$

3. Eliminate K:

$$m_K(c,s) = \sum_k p(k)p(c|k)p(s|k)$$

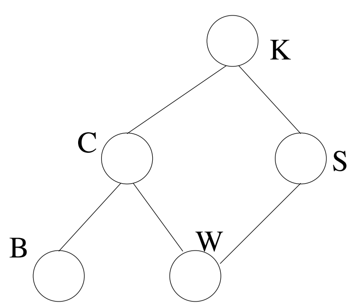
Example continued



$$m_K(c,s) = \sum_k p(k)p(c|k)p(s|k)$$

- Conceptually, at this point, we have *eliminated node K* from the graph
- The two nodes that create the new factor can be seen as *linked through an edge*

Example: Variable elimination for undirected graphs



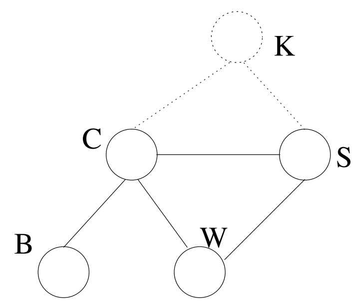
- When we eliminate K, we create a new factor:

$$m_K(c,s) = \sum_k \psi(k)\psi(k,c)\psi(k,s)$$

- From the point of view of graph operations, we:
  1. Connected the neighbors of K
  2. Eliminated K from the graph

Example continued

The new graph looks as follows:



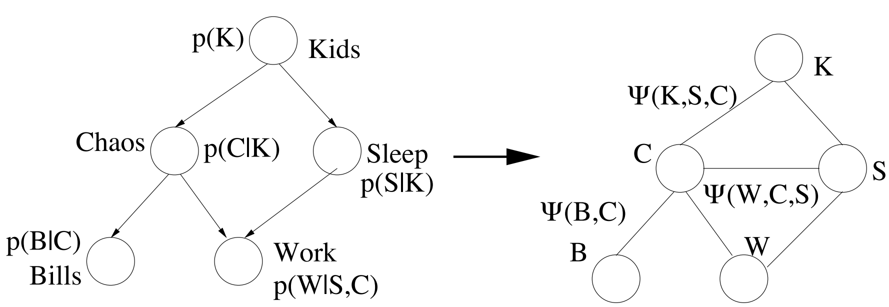
Variable elimination as node elimination: Undirected graphs

- A node will share potentials across cliques
- Hence, by summing out we create a factor which involves potentially *all its neighbors*
- Eliminating node X_i can be viewed as a two-step graph operation:
  1. Connect *all* neighbors of X_i (pairwise)
     This will make them all part of a clique (and the new factor is associated with this clique)
  2. Remove X_i (by summing out or by conditioning on its value)
- The resulting cliques are called **elimination cliques**
- The original graph together with all the added edges becomes a *triangulated graph* (every cycle of length > 3 has a chord)

Variable elimination as node elimination: Directed graphs

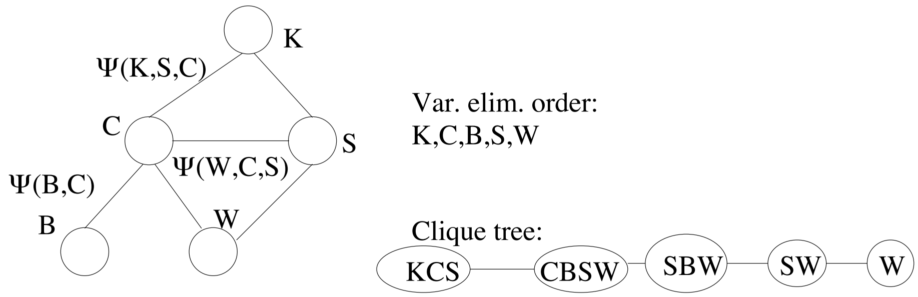
- The parameters of the graph are $p(x_i|x_{\pi_i})$.
- Hence, when we eliminate x_i, its parents will be involved in the same factor, *even if they did not share an edge before*
- To think of variable elimination as node elimination we must:
  1. Moralize the graph (marry all parents of common children and drop arc directions)
  2. Do elimination in the resulting undirected graph as before

Example



$$\psi(K,C,S) = p(K)p(C|K)p(S|K)$$
$$\psi(B,C) = p(B|C)$$
$$\psi(C,S,W) = p(W|C,S)$$

Example: Variable elimination



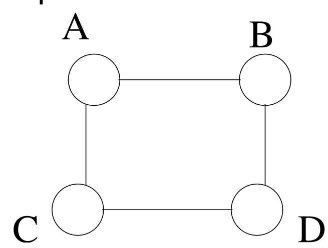
- We create a clique by connecting all the nodes that are involved in creating a factor (they would form a clique after elimination)
- The resulting structure is called a **clique tree**
- In general, a clique tree is a singly connected graph in which nodes are cliques of an underlying graph

Separator sets

- A **separator set** is the intersection of two corresponding cliques
- The separator sets are themselves cliques
- They provide an explicit representation of the *intermediate factors* that pass between cliques
- **Junction tree property:** the cliques containing a particular node form a *connected subtree*

Example: Junction tree property

The cliques containing a particular node form a *connected subtree*



How do we obtain a junction tree from this graph?

Constructing a junction tree

- Moralize the graph (if directed)
- Choose a node ordering and find the cliques generated by variable elimination. This gives a *triangulation* of the graph
- If the graph is not triangulated, we may not be able to get a clique tree with the junction tree property