Machine Intelligence 1

3.2 Bayesian Networks

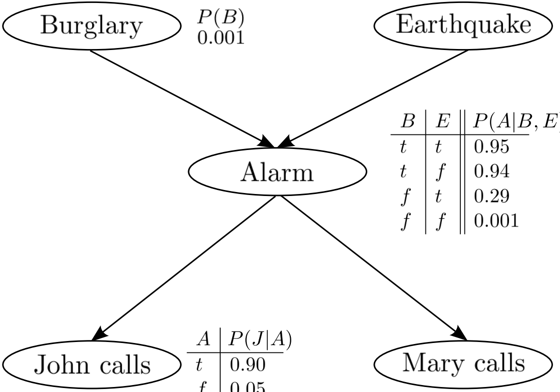
Prof. Dr. Klaus Obermayer

Fachgebiet Neuronale Informationsverarbeitung (NI)

WS 2017/2018

3.2.1 Directed Acyclic Graphs

A "Californian" example



$P(B)$
0.001

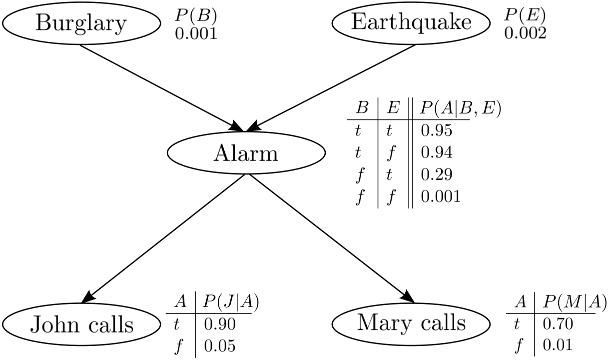
$P(E)$
0.002

B	E	$P(A\|B,E)$
t	t	0.95
t	f	0.94
f	t	0.29
f	f	0.001

A	$P(J\|A)$
t	0.90
f	0.05

A	$P(M\|A)$
t	0.70
f	0.01

A "Californian" example



$P(B)$
0.001

$P(E)$
0.002

B	E	$P(A\|B,E)$
t	t	0.95
t	f	0.94
f	t	0.29
f	f	0.001

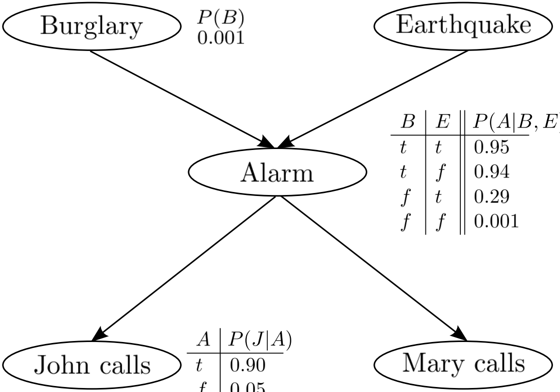
A	$P(J\|A)$
t	0.90
f	0.05

A	$P(M\|A)$
t	0.70
f	0.01

- set of random variables $\rightsquigarrow$ nodes of the graph
- direct influences between variables $\rightsquigarrow$ directed links between nodes
- nodes x_i are annotated with the conditional probabilities

  $P(X_i \,|\, \text{parents}(X_i))$

A "Californian" example



$P(B)$
0.001

$P(E)$
0.002

B	E	$P(A\|B,E)$
t	t	0.95
t	f	0.94
f	t	0.29
f	f	0.001

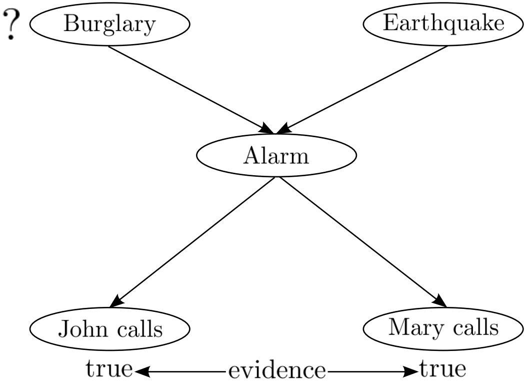
A	$P(J\|A)$
t	0.90
f	0.05

A	$P(M\|A)$
t	0.70
f	0.01

$$
\begin{aligned}
P(J,M,A,B,E) &= P(J|M,A,B,E)\,P(M|A,B,E)\,P(A|B,E)\,P(B|E)\,P(E) \\
&= P(J|A)\,P(M|A)\,P(A|B,E)\,P(B)\,P(E)
\end{aligned}
$$

Inference

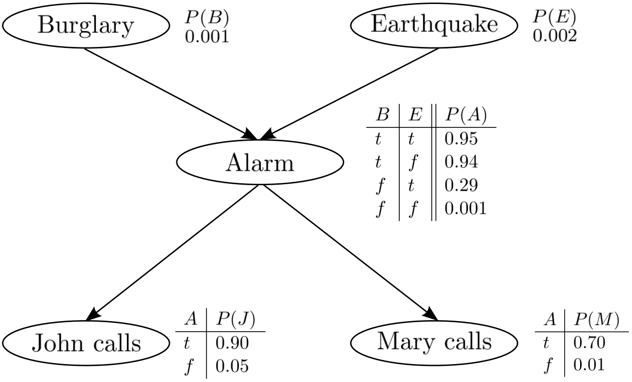
- Both Mary and John are calling



- Was there a burglary?

Inference

- Both Mary and John are calling ($M =$ true and $J =$ true)



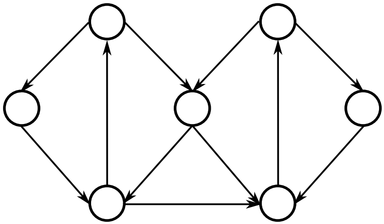
B	E	$P(A)$
t	t	0.95
t	f	0.94
f	t	0.29
f	f	0.001

A	$P(J)$
t	0.90
f	0.05

A	$P(M)$
t	0.70
f	0.01

- Was there a burglary?: $P(B \mid M = \text{true} \wedge J = \text{true})$ (see blackboard)

Directed graphs

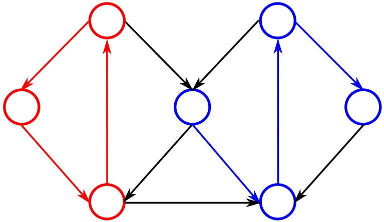


Directed graph $G = (V, K)$

$V \;\rightarrow\;$ set of nodes

$K \;\rightarrow\;$ set of directed edges

Directed graphs



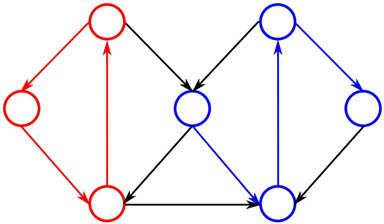
Directed graph $G = (V, K)$

$V \;\rightarrow\;$ set of nodes

$K \;\rightarrow\;$ set of directed edges

- **path:** sequence $\{x_i \in V\}_{i=1}^{n}$ with $(x_i, x_{i+1}) \in K$
- **cycle:** path with $x_1 = x_{n+1}$,

Directed graphs



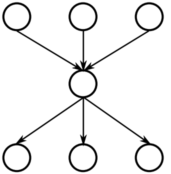
Directed graph $G = (V, K)$

$V \;\rightarrow\;$ set of nodes

$K \;\rightarrow\;$ set of directed edges

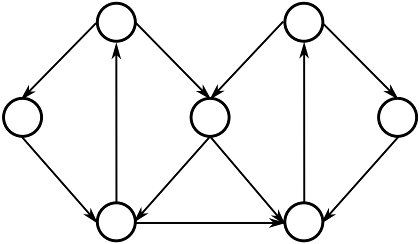
- **path:** sequence $\{x_i \in V\}_{i=1}^{n}$ with $(x_i, x_{i+1}) \in K$
- **cycle:** path with $x_1 = x_{n+1}$,

- **parents** of x_i: $\{x_j \,|\, (x_j, x_i) \in K\}$
- **children** of x_i: $\{x_j \,|\, (x_i, x_j) \in K\}$

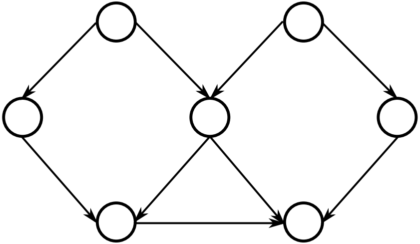


Directed acyclic graphs (DAGs)

- DAG: Directed graph which does not contain cycles.



directed graph with cycles



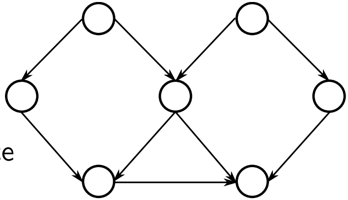
directed graph without cycles

DAG and distributions

- a DAG corresponds to a factorization of the joint probability

$$P(X_1, \ldots, X_n) \quad = \quad \prod_{i=1}^{n} P\big(X_i \,|\, parents(X_i)\big)$$

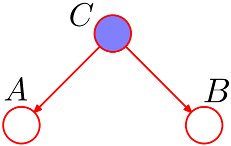
- efficient representation of statistical dependencies
  - topology: qualitative relationships
  - annotation: quantitative relationships
- **not** an efficient representation for inference



Conditional independence

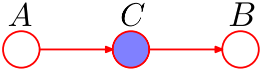
- Node $A \perp X_i$ is **conditionally independent** of all nodes X_i given its Markov blanket.
- **Markov blanket** of a node A: parents, children, and children's parents.
- Simple examples:

$C \in parents(A)$,



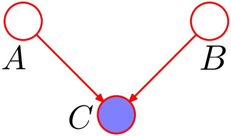
$A \perp B \mid C$

$C \in children(A)$,



$A \perp B \mid C$

$B \in parents(children(A))$

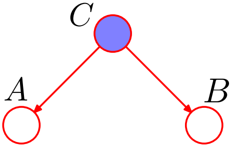


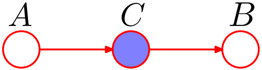
$A \not\perp B \mid C$

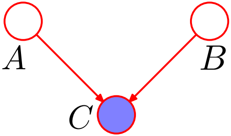
Conditional independence vs. statistical independence

- Node $A \perp X_i$ is **conditionally independent** of all nodes X_i given its Markov blanket.
- **Markov blanket** of a node A: parents, children, and children's parents.
- Simple examples:

$C \in parents(A)$, $C \in children(A)$, $B \in parents(children(A))$

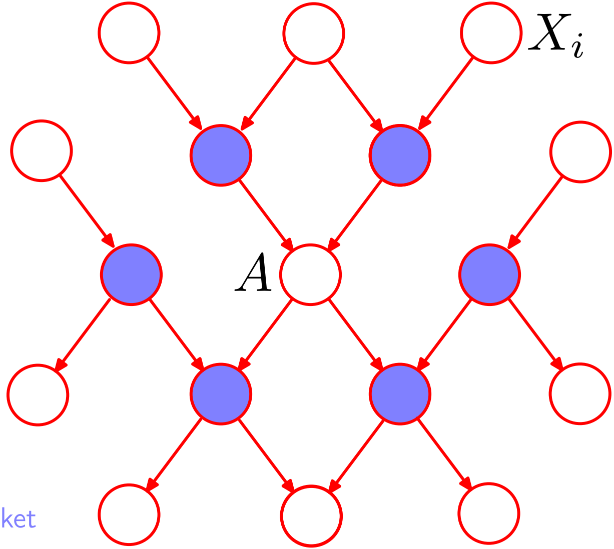






$A \perp B \mid C$	$A \perp B \mid C$	$A \not\perp B \mid C$
$A \not\perp B \mid \emptyset$	$A \not\perp B \mid \emptyset$	$A \perp B \mid \emptyset$

Markov blanket



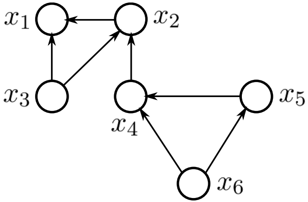
Markov blanket

Topological sorting

- Factorization of the unconditional probability:

$$P(X_1, \ldots, X_n) \quad = \quad \prod_{i=1}^{n} P(X_i \,|\, parents(X_i))$$

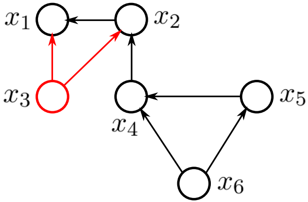
- Topological sorting: From parents to kids:
  1. select and queue a node without parents
  2. delete that node from the DAG
  3. repeat until DAG is empty



- Edges are always directed from nodes with lower to nodes with higher indices.

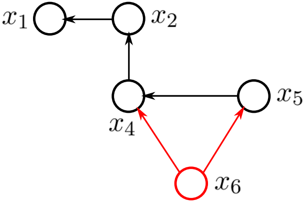
Topological sorting

- example:
  - x_3,



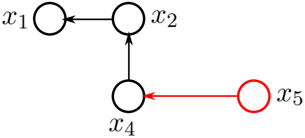
Topological sorting

- example:
  - $x_3, x_6,$



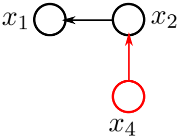
Topological sorting

- example:
  - $\color{red}{x_3}, x_6, x_5,$



Topological sorting

- example:
  - $x_3, x_6, x_5, x_4,$



Topological sorting

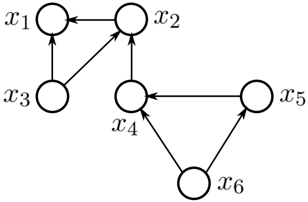
- example:
  - $\color{red}{x_3}, x_6, x_5, x_4, x_2,$

Topological sorting

- example:
  - $x_3, x_6, x_5, x_4, x_2, x_1$

Topological sorting

- example:
  - $x_3, x_6, x_5, x_4, x_2, x_1$
- other possible topological orderings:
  - $x_6, x_5, x_4, x_3, x_2, x_1$
  - $x_6, x_5, x_3, x_4, x_2, x_1$
  - $x_6, x_3, x_5, x_4, x_2, x_1$

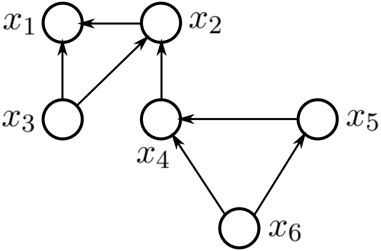


3.2.2 Inference on Bipartite Trees

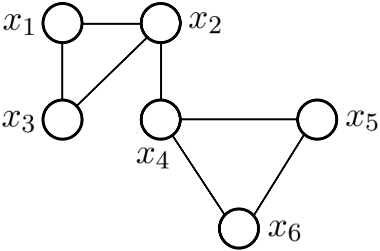
Undirected graphs

- **undirected graph:** directed graph with symmetric (undirected) edges

$$(x_i, x_j) \in K \quad \Rightarrow \quad (x_j, x_i) \in K$$



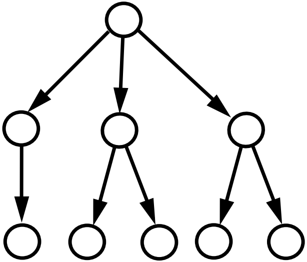
directed graph



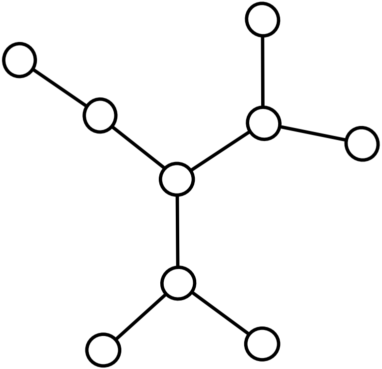
undirected graph

Trees

- **tree**: graph where each existing path between nodes is *unique*



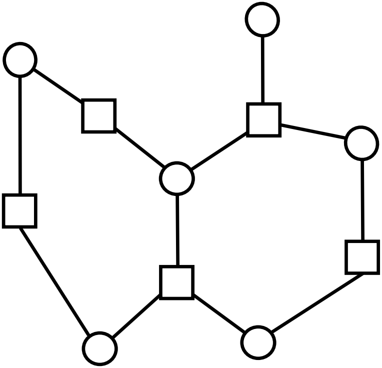
directed tree



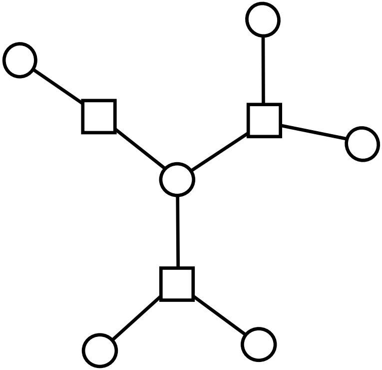
undirected tree

Bipartite graphs

- **bipartite** graph: two *types* of nodes (□ and ◯)
  - each type can only connect to the other (□—◯)

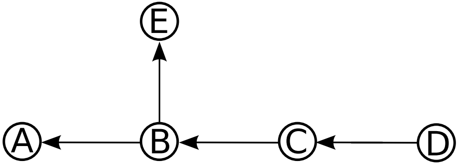


bipartite graph



bipartite tree

Tree-shaped DAGs

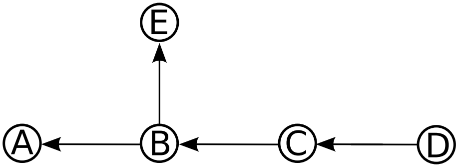


$$P(A,B,C,D,E) \quad = \quad P(A|B)\,P(E|B)\,P(B|C)\,P(C|D)\,P(D)$$

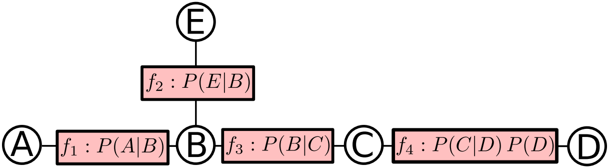
- **task**: evidence e for the value of E, update belief in A, i.e. $P(A|E{=}e)$

$$P(A|E{=}e) = \frac{P(A,E{=}e)}{P(E{=}e)} = \overbrace{\alpha\,P(A,E{=}e)}^{\text{normalization}} = \overbrace{\alpha \sum_{B,C,D} P(A,B,C,D,E{=}e)}^{\text{marginalization}}$$

The bipartite tree

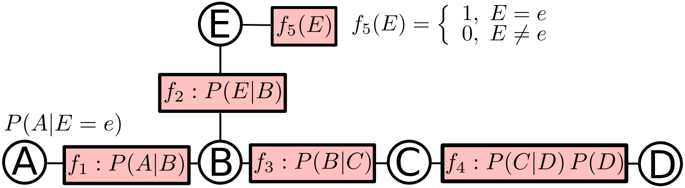


$$P(A,B,C,D,E) \quad = \quad \underbrace{P(A|B)}_{f_1(\underbrace{A,B}_{\phi_1})}\,\underbrace{P(E|B)}_{f_2(\underbrace{B,E}_{\phi_2})}\,\underbrace{P(B|C)}_{f_3(\underbrace{B,C}_{\phi_3})}\,\underbrace{P(C|D)\,P(D)}_{f_4(\underbrace{C,D}_{\phi_4})} \quad = \quad \prod_{k=1}^{4} f_k(\phi_k)$$



Inference

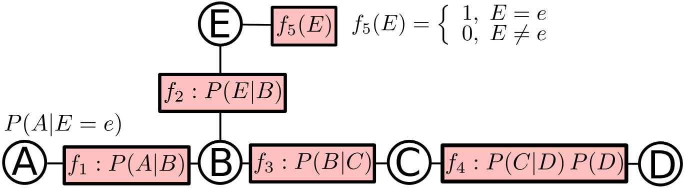
- Computation of $P(A|E = e)$



$$
\begin{aligned}
P(A|E = e) &= \alpha \sum_{B,C,D} P(A, B, C, D, E = e) = \alpha \sum_{B,C,D,E} \prod_{k=1}^{5} f_k(\phi_k) \\
&= \alpha \sum_{B,C,D,E} P(A|B)\, P(E|B)\, P(B|C)\, P(C|D)\, P(D)\, f_5(E)
\end{aligned}
$$

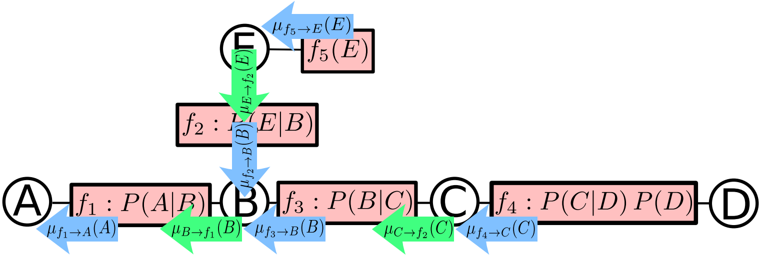
Inference

- Computation of $P(A|E = e)$



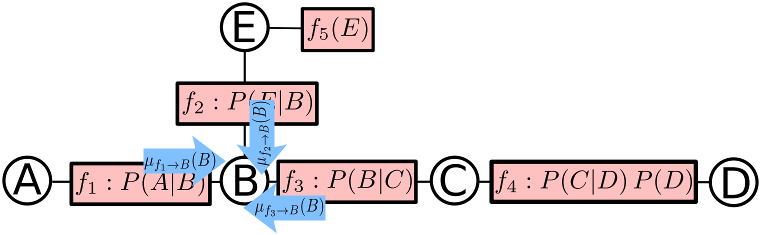
$$P(A|E = e) \quad = \quad \alpha \sum_B P(A|B) \Big(\sum_E P(E|B)\, f_5(E) \Big) \Big(\sum_C P(B|C) \sum_D P(C|D)\, P(D) \Big)$$

Message passing



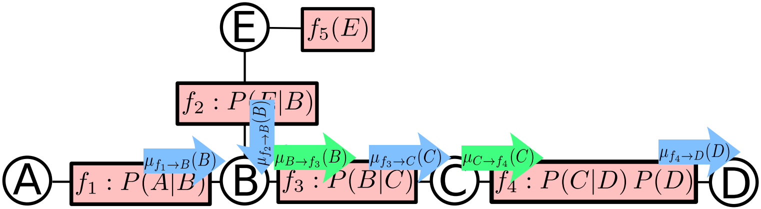
$$P(A|E=e) \;=\; \underbrace{\alpha\sum_B P(A|B)\,\overbrace{\underbrace{\left(\sum_E P(E|B)\,\overbrace{\underbrace{f_5(E)}_{\mu_{f_5\to E}(E)}}^{\mu_{E\to f_2}(E)}\right)}_{\mu_{f_2\to B}(B)}\underbrace{\left(\sum_C P(B|C)\,\overbrace{\underbrace{\sum_D P(C|D)\,P(D)}_{\mu_{f_4\to C}(C)}}^{\mu_{C\to f_3}(C)}\right)}_{\mu_{f_3\to B}(B)}}^{\mu_{B\to f_1}(B)}}_{\mu_{f_1\to A}(A)}$$

Message passing



$$P(B|E=e) = \alpha \underbrace{\sum_A P(A|B)}_{\mu_{f_1 \to B}(B)} \underbrace{\sum_E P(E|B)\, f_5(E)}_{\mu_{f_2 \to B}(B)} \underbrace{\sum_C P(B|C) \sum_D P(C|D)\, P(D)}_{\mu_{f_3 \to B}(B)}$$

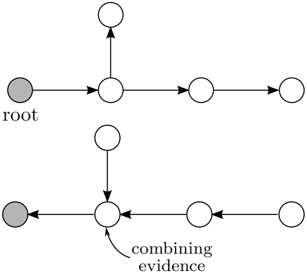
Message passing



$$P(D|E=e) \;=\; \alpha \underbrace{\sum_C P(C|D)\, P(D) \underbrace{\overbrace{\sum_B P(B|C) \overbrace{\underbrace{\sum_E P(E|B)\, f_5(E)}_{\mu_{f_2\to B}(B)} \underbrace{\sum_A P(A|B)}_{\mu_{f_1\to B}(B)}}^{\mu_{B\to f_3}(B)}}^{\mu_{C\to f_4}(C)}}_{\mu_{f_3\to C}(C)}}_{\mu_{f_4\to D}(D)}$$

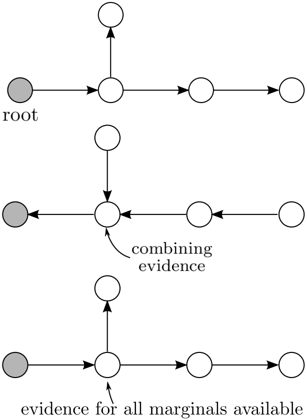
Message passing

- first pass from root to leaves: “request”
- second pass from leaves to root: “collect”

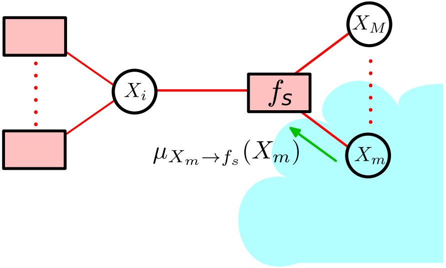


Message passing

- first pass from root to leaves: “request”
- second pass from leaves to root: “collect”
- a third pass can calculate all other marginals: “distribute”



The sum-product algorithm

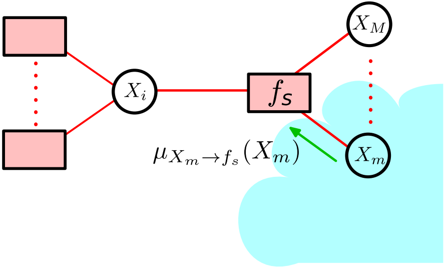


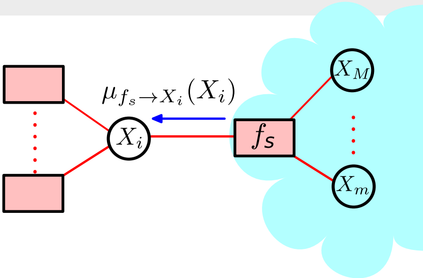
- product message from X_m to f_s

$$\mu_{X_m \to f_s}(X_m) := \prod_{l \in \text{neighbor}(X_m) \setminus \{f_s\}} \mu_{f_l \to X_m}(X_m) \qquad \text{(product)}$$

Bishop 2006 (p. 402)

The sum-product algorithm





- product message from X_m to f_s
- sum message from f_s to X_i

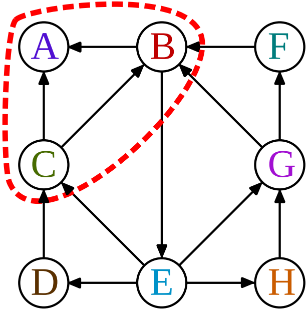
$$\mu_{X_m \to f_s}(X_m) := \prod_{l \in \text{neighbor}(X_m) \setminus \{f_s\}} \mu_{f_l \to X_m}(X_m) \qquad \text{(product)}$$

$$\mu_{f_s \to X_i}(X_i) := \sum_{X_m, \dots, X_M} f_s(X_i, X_m, \dots, X_M) \prod_{k \in \text{neighbor}(f_s) \setminus \{X_i\}} \mu_{X_k \to f_s}(X_k) \qquad \text{(sum)}$$

Bishop 2006 (p. 402)

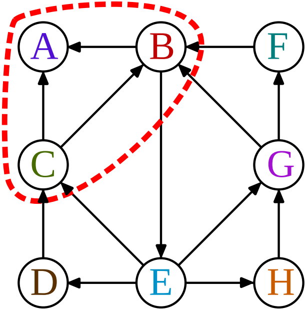
Junction trees

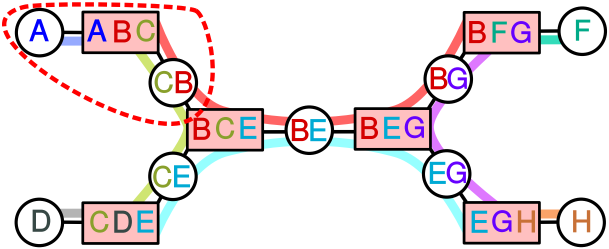
- efficient inference requires trees
- DAGs may not be trees
  - hide sets of nodes which violate the tree property within **cliques**



Junction trees

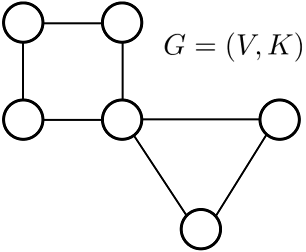
- efficient inference requires trees
- DAGs may not be trees
  - hide sets of nodes which violate the tree property within **cliques**
- construct a tree based on cliques



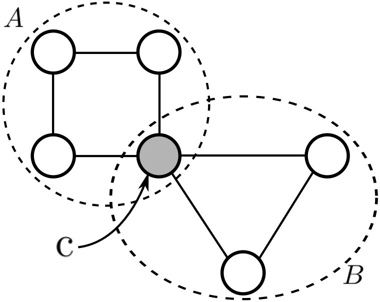


3.2.3 Decomposable Undirected Graphs

Separators



undirected graph G



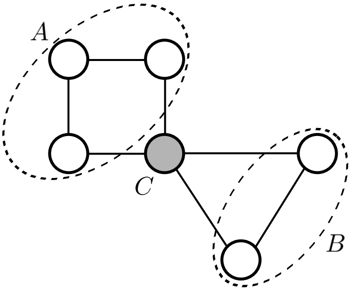
node c separates subsets A and B

Separator

A set C **separates** two undirected subgraphs A and B if *every* path from A to B has to pass through an element of C.

Decomposable graphs

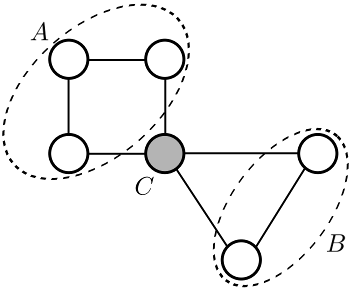
- A, B, C are a **proper decomposition** of an undirected graph $G = (V, K)$ if:
  - A, B, C are non-empty and disjoint subsets with $V = A \cup B \cup C$,
  - C separates A and B, and
  - C is complete.



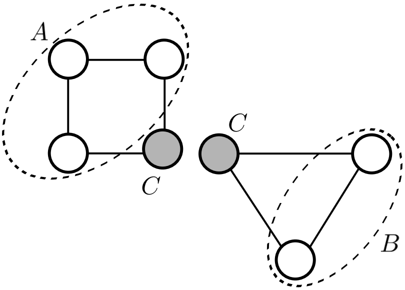
a proper decomposition

Decomposable graphs

- A, B, C are a **proper decomposition** of an undirected graph $G = (V, K)$ if:
  - A, B, C are non-empty and disjoint subsets with $V = A \cup B \cup C$,
  - C separates A and B, and
  - C is complete.
- G is **decomposable** if it is complete, or a proper decomposition A, B, C exist, where $G_{A \cup C}$ and $G_{B \cup C}$ are decomposable.



not a decomposable graph

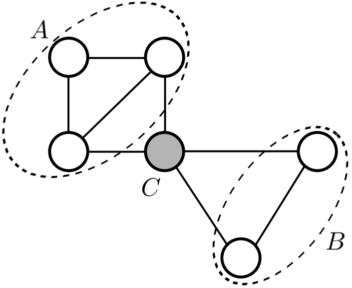


A is not complete

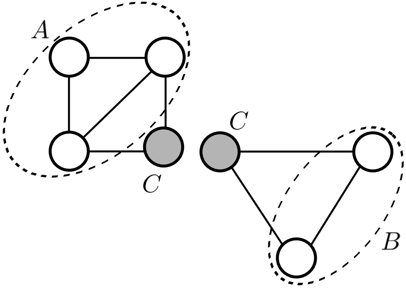
Decomposable graphs

- A, B, C are a **proper decomposition** of an undirected graph $G = (V, K)$ if:
  - A, B, C are non-empty and disjoint subsets with $V = A \cup B \cup C$,
  - C separates A and B, and
  - C is complete.

- G is **decomposable** if it is complete, or a proper decomposition A, B, C exist, where $G_{A\cup C}$ and $G_{B\cup C}$ are decomposable.



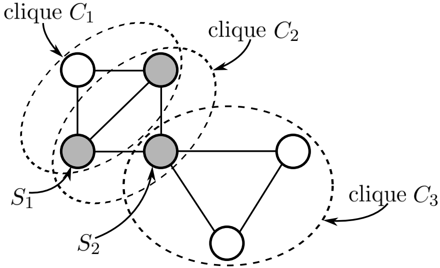
a decomposable graph



A and B are complete

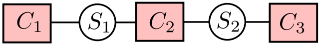
Cliques and separators

- **cliques** are maximally complete subgraphs
- decomposable graphs can be decomposed into cliques and separators
- cliques and separators form a bipartite graph

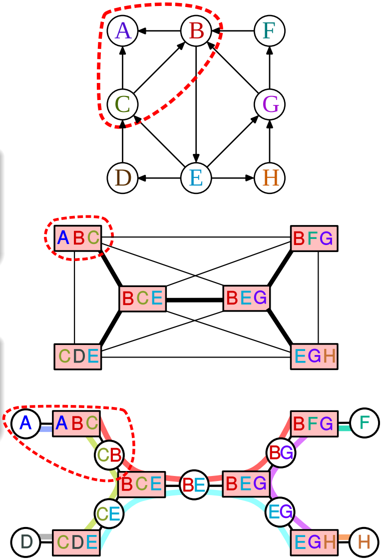


decomposable graph

$\rightsquigarrow$



Junction trees



Existence (Cowell et al., 1999)

There exist a junction tree of cliques for the graph $\mathcal{G}$ if and only if $\mathcal{G}$ is **decomposable**.

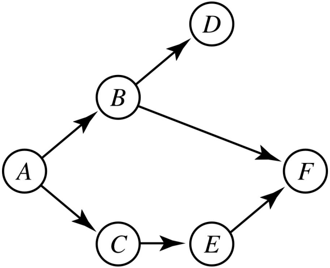
Running intersection property

All nodes on the path between two cliques, which both contain variable X, also contain X.

3.2.4 Construction of the Junction Tree

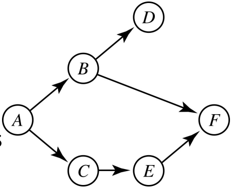
Define the knowledge base

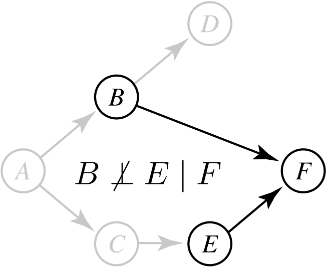
1. construct the DAG

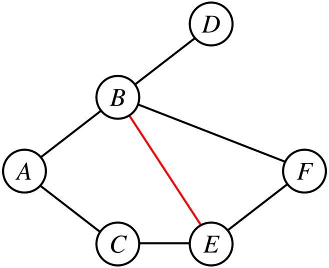


Construction of the undirected decomposable graph

1. construct the DAG
2. convert to the moral graph
   - undirected edges represent conditional dependence
   - insert undirected edges between all parents of nodes
   - convert all directed to undirected edges

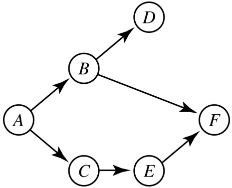


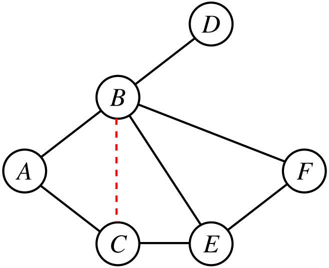


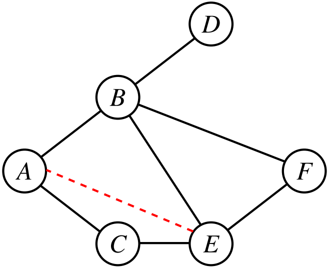


Construction of the undirected decomposable graph

1. construct the DAG
2. convert to the moral graph
3. construct a chordal (decomposable) graph
   - add chords to all circles of length 4+ ("shortcuts")
   - chordal (decomposable) graphs not unique

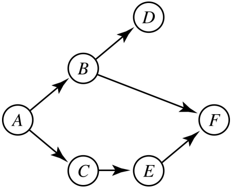


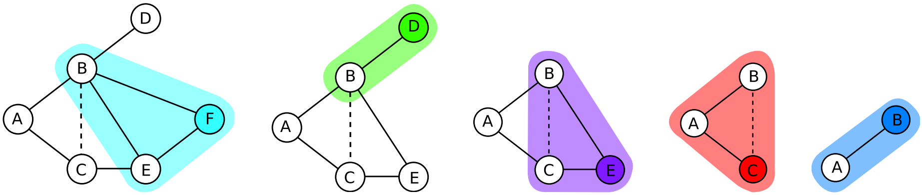




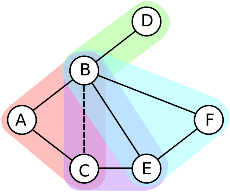
Identification of cliques and separators

1. construct the DAG
2. convert to the moral graph
3. construct a chordal (decomposable) graph
4. identify cliques
   - cliques are maximally complete subgraphs
   - cliques can be found by elimination
   - requires variable ordering by topological sorting: F, D, E, C, B, A

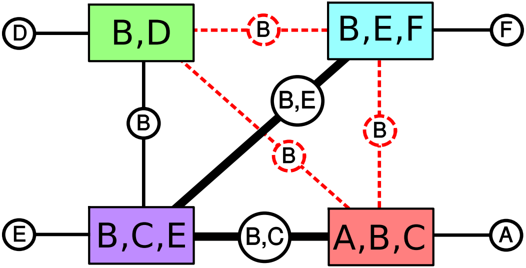




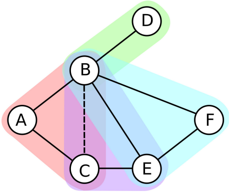
Construction of the junction tree



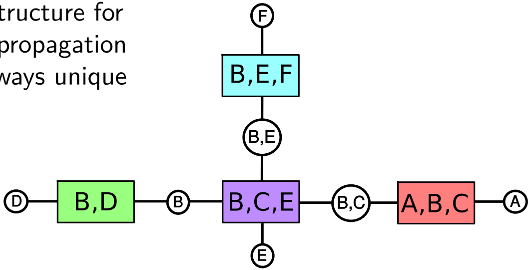
1. construct the DAG
2. convert to the moral graph
3. construct a chordal (decomposable) graph
4. identify cliques
5. construct bipartite graph
   - edges weighted by separator size $\rightarrow$ number of nodes within separator
   - find a *maximal spanning tree*



Construction of the junction tree



1. construct the DAG
2. convert to the moral graph
3. construct a chordal (decomposable) graph
4. identify cliques
5. construct bipartite graph
6. junction tree for inference
   - maximal spanning tree of the clique graph
   - data structure for belief propagation
   - not always unique

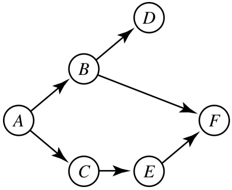


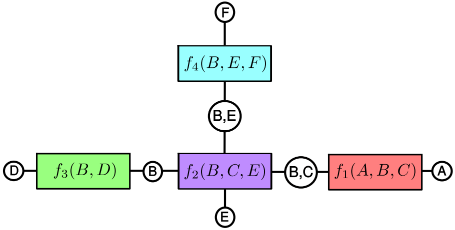
Inference

1. initialization of the clique potentials $f_k(\mathcal{X}_k)$
   - in the order established by topological sorting (e.g. $A \to B \to C \to E \to D \to F$)

$f_1(A,B,C)$	$f_2(B,C,E)$	$f_3(B,D)$	$f_4(B,E,F)$
$P(C\|A)\,P(B\|A)\,P(A)$	$P(E\|C)$	$P(D\|B)$	$P(F\|B,E)$

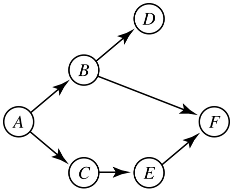
$$P(A,B,C,D,E,F) \quad = \quad P(A)\,P(B|A)\,P(C|A)\,P(D|B)\,P(E|C)\,P(F|B,E)$$

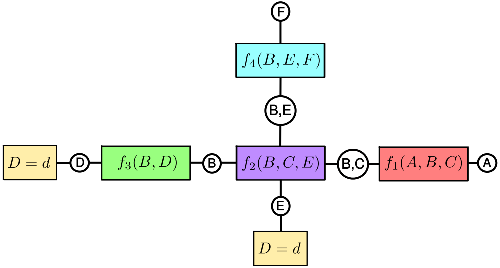




Inference

1. initialization of the clique potentials $f_k(\mathcal{X}_k)$
2. modification of the clique potentials by the observed evidence
   - for each observation $Y = y$ find *one* f_k with $Y \in \mathcal{X}_k$
   - add a separator node $f_k(Y) = \begin{cases} 1, & Y = y \\ 0, & Y \neq y \end{cases}$
   - example: $D = d$ and $E = e$

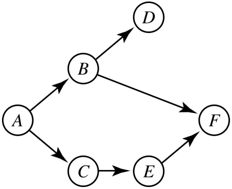


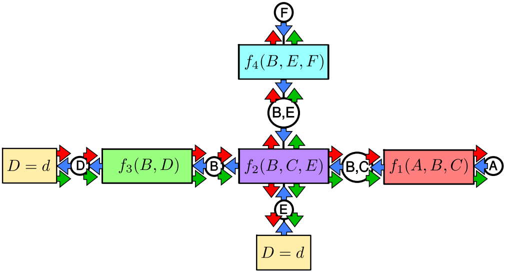


Inference

1. initialization of the clique potentials $f_k(\mathcal{X}_k)$
2. modification of the clique potentials by the observed evidence
3. message passing
   - begin "request" pass from arbitrary node N, e.g. f_1
   - wait for all message of "collect" pass to return
   - send last "distribute" pass from N

   $\Rightarrow$ all marginals are computed simultaneously

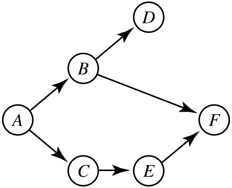


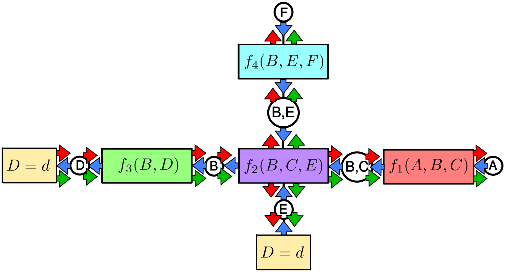


Inference

1. initialization of the clique potentials $f_k(\mathcal{X}_k)$
2. modification of the clique potentials by the observed evidence
3. message passing
4. calculate marginals from messages

$$P(C \,|\, D = d \wedge E = e) \quad = \quad \sum_{B} \mu_{f_1 \to BC}(B, C) \cdot \mu_{f_2 \to BC}(B, C)$$





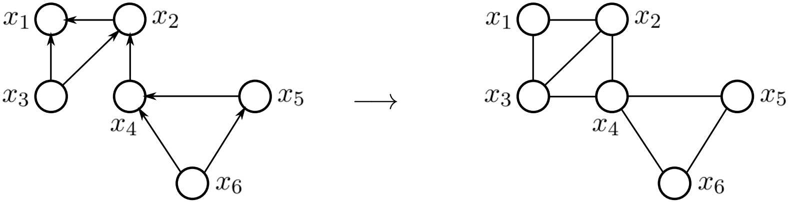
End of Section 3.2

the following slides contain

OPTIONAL MATERIAL

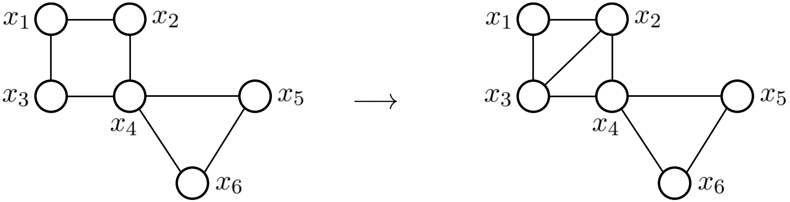
Moral graph

- **moral graphs** are the undirected equivalent of DAGs
  - connect nodes x_i of a DAG to their Markov blanket
  - insert undirected edges to all parents of all children of x_i
  - convert all directed to undirected edges



Chordal graph

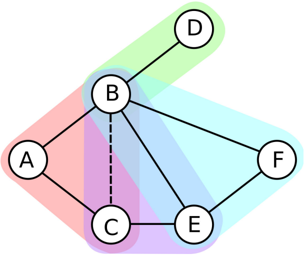
- **chordal graphs** become decomposable by inserting edges
    - insert undirected edges to all circles with 4 or more nodes



- transform DAG into chordal graph $\rightarrow$ junction tree

Decomposable graphs and distributions

- $A \perp B \,|\, C \Leftrightarrow A, B, C$ is proper decomposition of G
- decomposable graph factors into marginal distributions



$$P(\underline{\mathbf{x}}) = \frac{\prod\limits_{\text{cliques } C} P_C(\underline{\mathbf{x}}_C)}{\prod\limits_{\text{separators } S} P_S(\underline{\mathbf{x}}_S)}$$

$$P(A,B,C,D,E,F) \;=\; \frac{\overbrace{P(A,B,C)\,P(B,D)\,P(B,C,E)\,P(B,E,F)}^{\text{cliques}}}{\underbrace{P(B)\,P(B,C)\,P(B,E)}_{\text{separators}}}$$