Message Passing
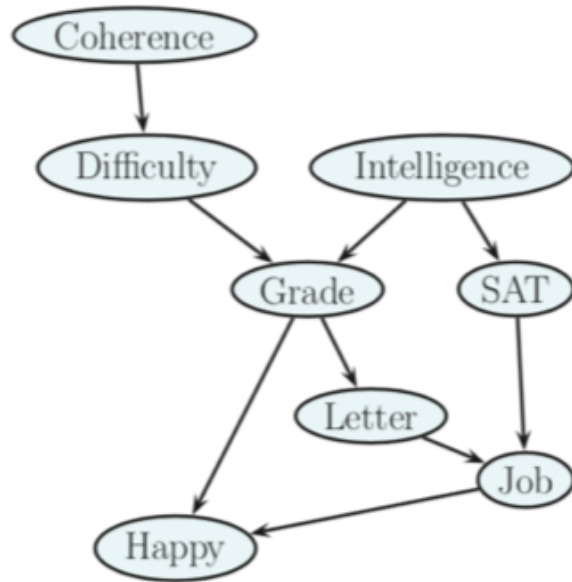# and
Junction Tree Algorithms

Kayhan Batmanghelich

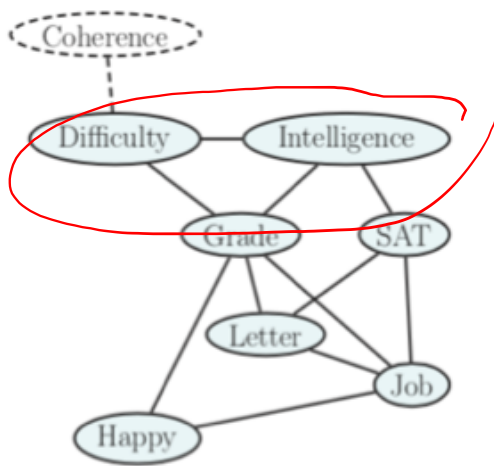# Review

$$P(C, D, I, G, S, L, J, H)$$
$$= \quad P(C)P(D|C)P(I)P(G|I, D)P(S|I)P(L|G)P(J|L, S)P(H|G, J)$$

$$p(C, D, I, G, S, L, J, H)$$
$$= \quad \psi_C(C)\psi_D(D, C)\psi_I(I)\psi_G(G, I, D)\psi_S(S, I)\psi_L(L, G)\psi_J(J, L, S)\psi_H(H, G, J)$$
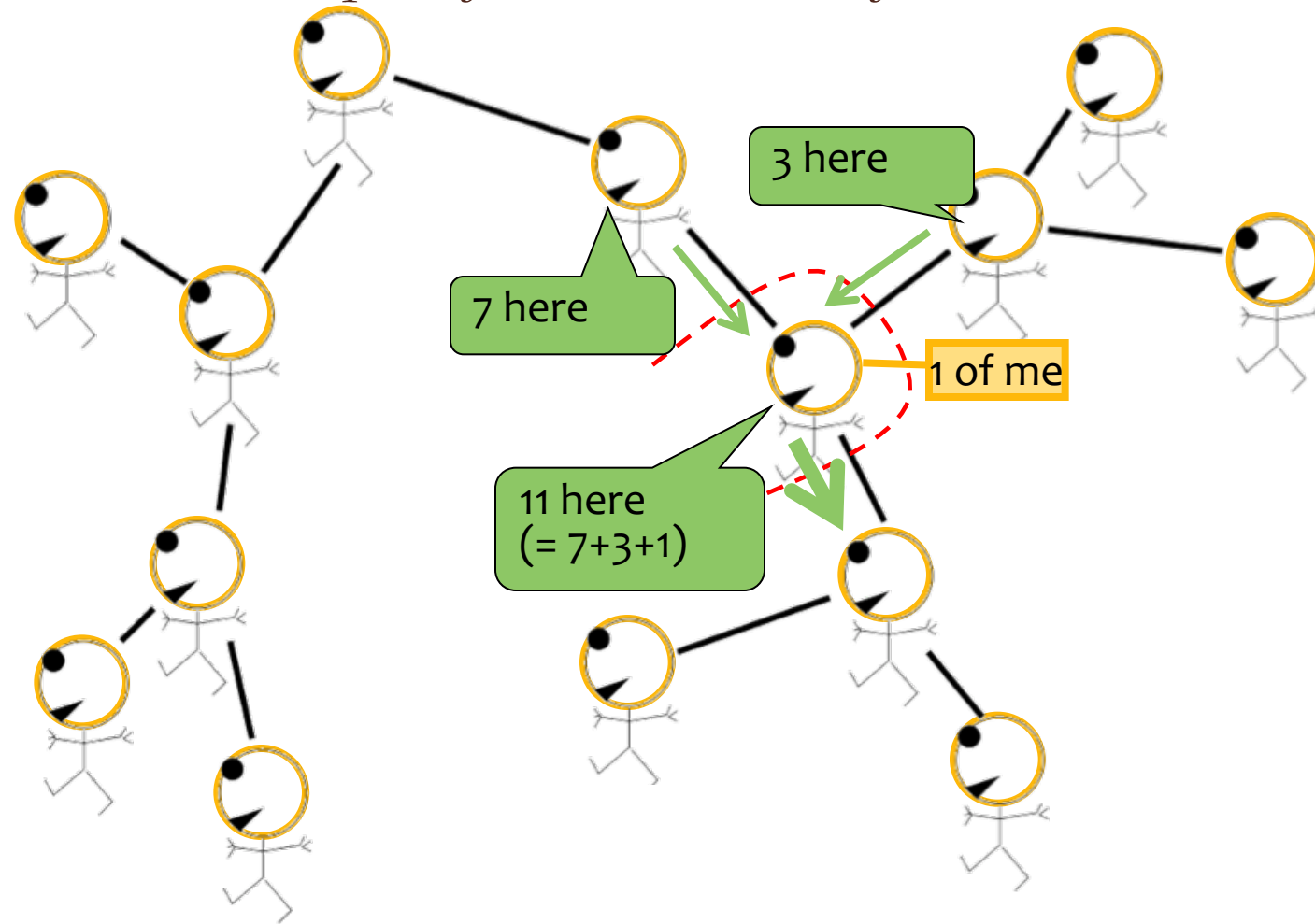
Review

$$(C, D, I, H, G, S, L)$$

$$\{C, D\}, \{D, I, G\}, \{G, L, S, J\}, \{G, J, H\}, \{G, I, S\}$$

Great Ideas in ML: Message Passing

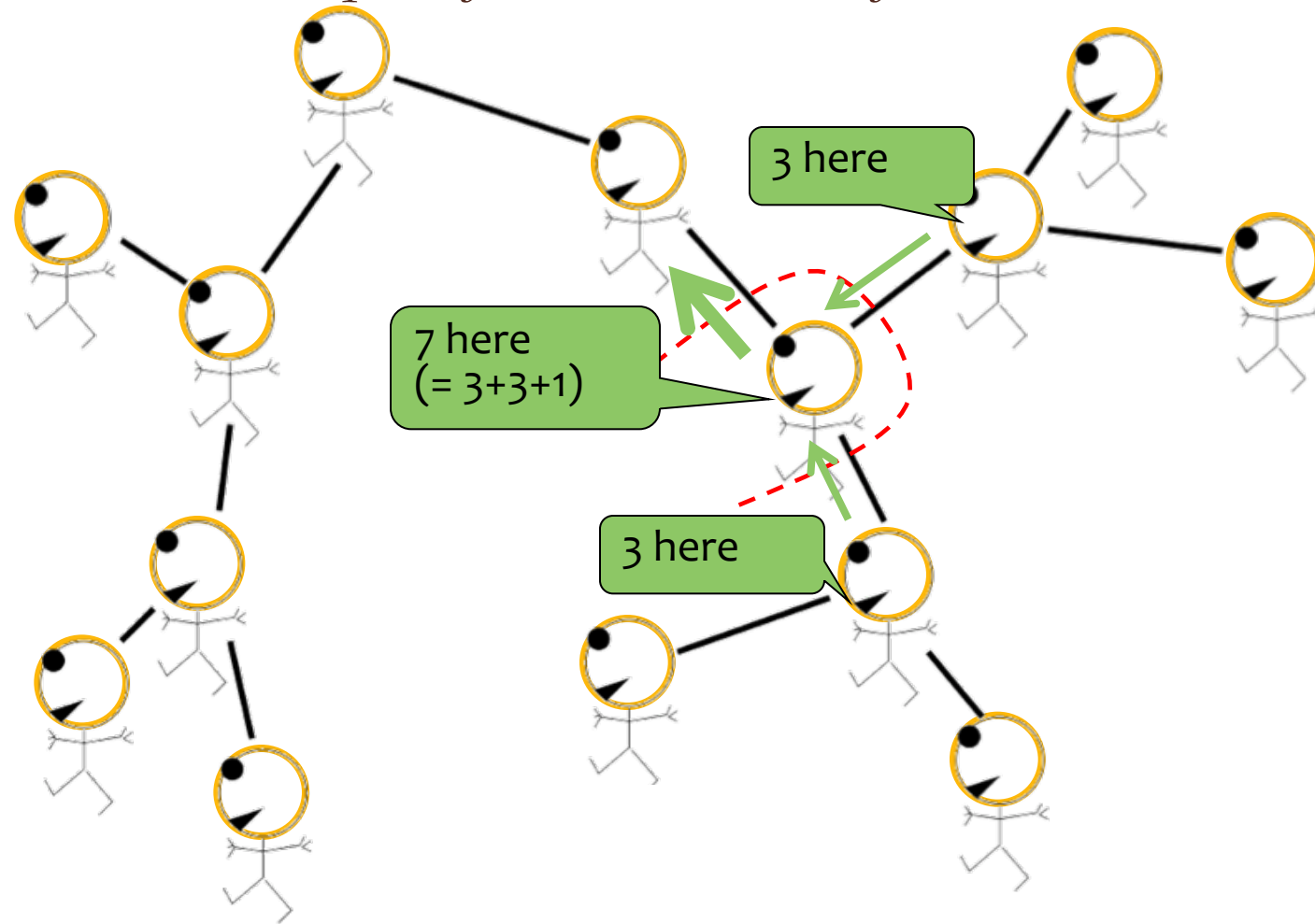*Each soldier receives reports from all branches of tree*

3 here

7 here

1 of me

11 here
(= 7+3+1)

adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing
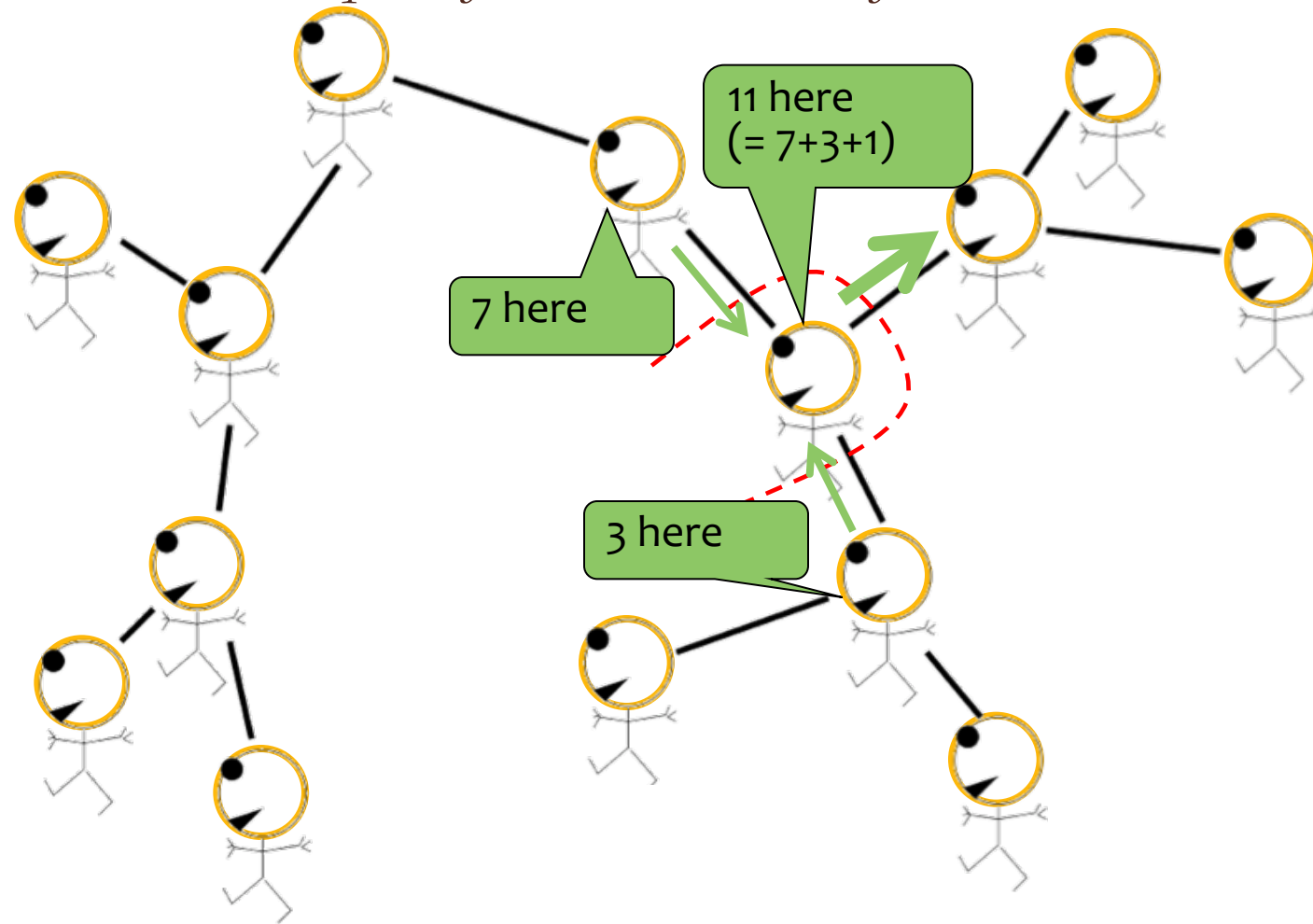
Each soldier receives reports from all branches of tree

3 here
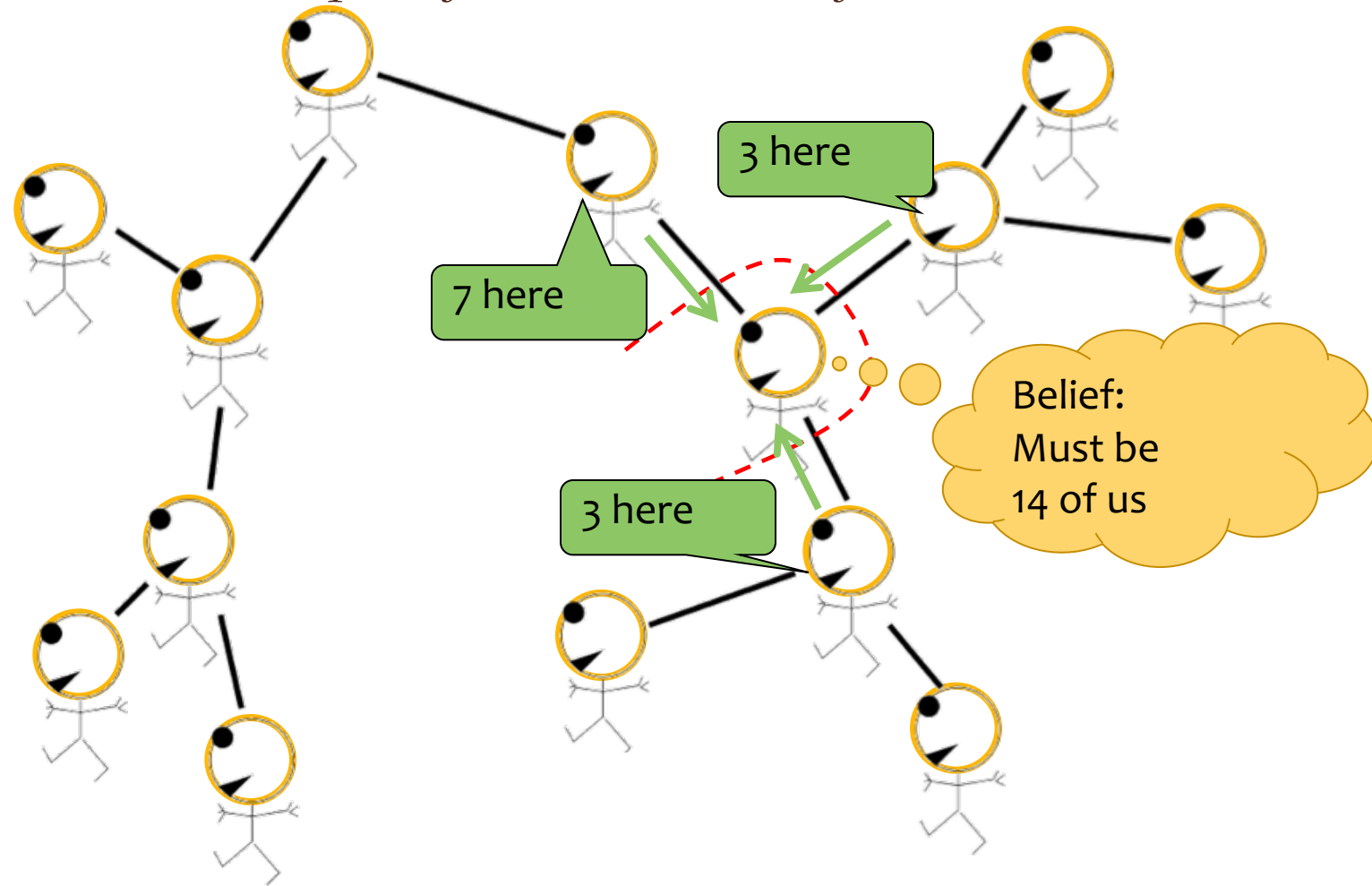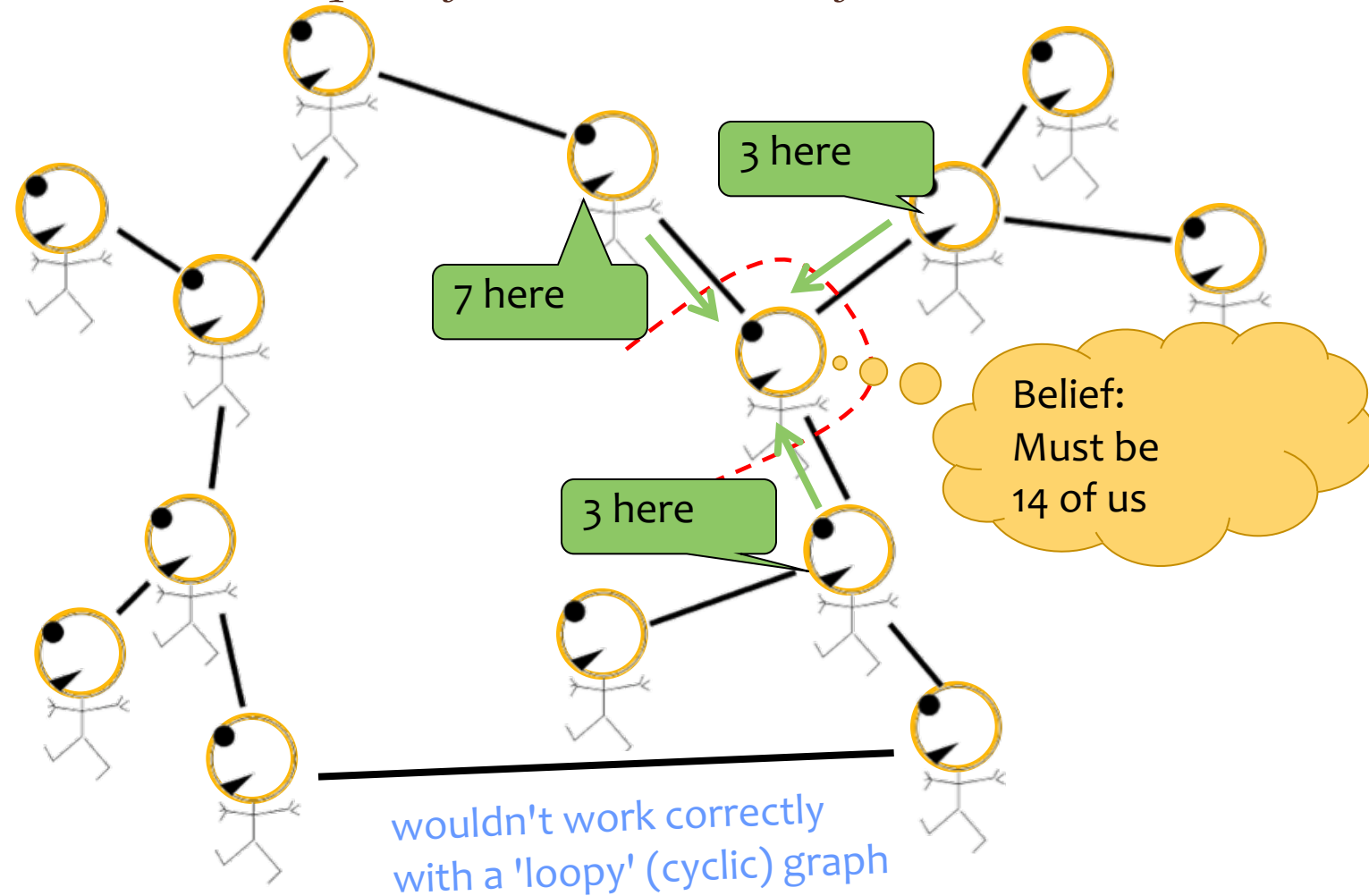
7 here

3 here

Belief:
Must be
14 of us

wouldn't work correctly
with a 'loopy' (cyclic) graph

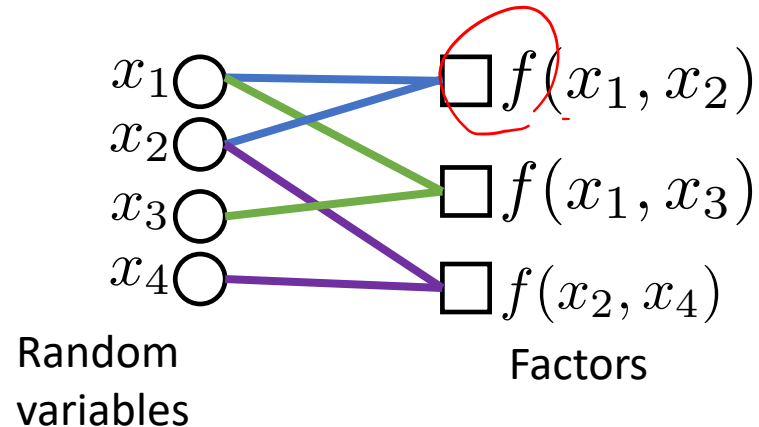adapted from MacKay (2003) textbook

Review

Message from one C1 to C2:
Multiply all incoming messages
with the local factor and **sum** over
variables that are not shared
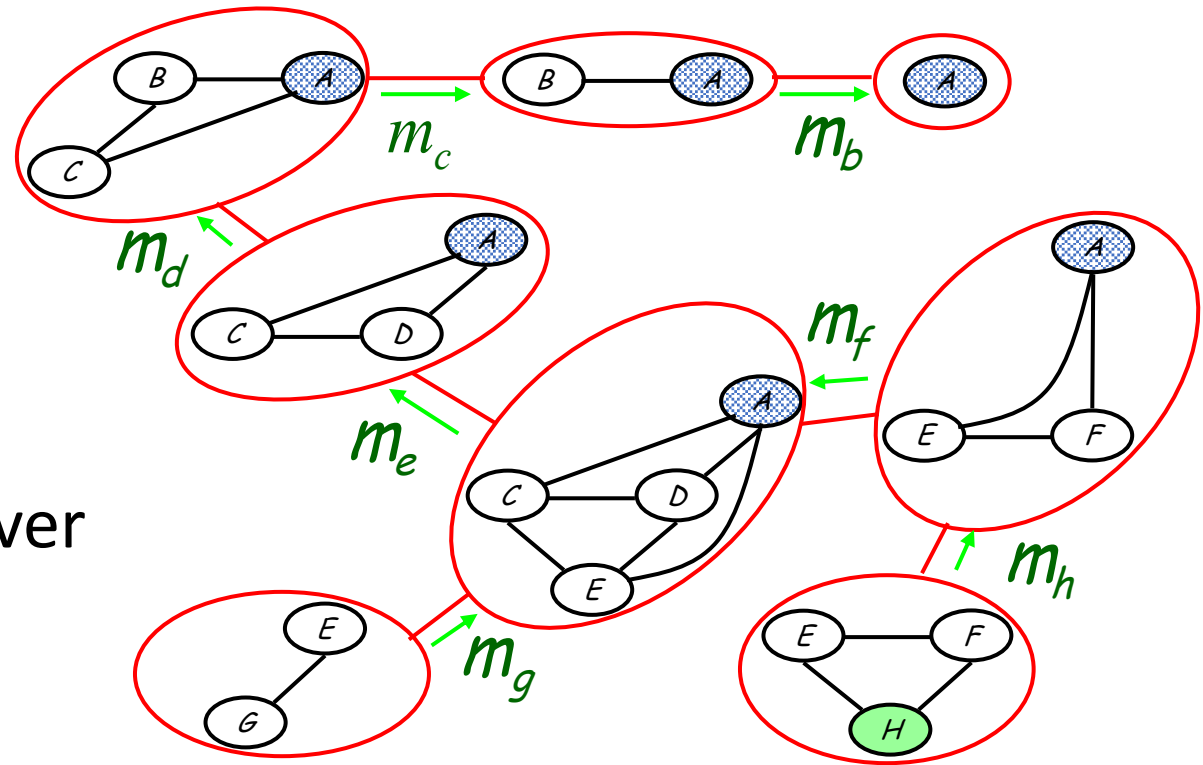
$$m_e(a,c,d)$$
$$= \sum_e p(e\,|\,c,d)m_g(e)m_f(a,e)$$

Message passing (Belief Propagation)
on
singly connected graph

Remember this: Factor Graph?

Random variables

Factors

x_1 x_2 x_3 x_4

$f(x_1, x_2)$

$f(x_1, x_3)$

$f(x_2, x_4)$

- A factor graph is a graphical model representation that unifies directed and undirected models

- It is an undirected bipartite graph with two kinds of nodes.
  - Round nodes represent variables,
  - Square nodes represent factors

and there is an edge from each variable to every factor that mentions it.

- We are going to study messages passing between nodes.

11

How General Are Factor Graphs?

- Factor graphs can be used to describe
  - Markov Random Fields (undirected graphical models)
    - i.e., log-linear models over a tuple of variables
  - Conditional Random Fields
  - Bayesian Networks (directed graphical models)

- *Inference* treats all of these interchangeably.
  - Convert your model to a factor graph first.
  - Pearl (1988) gave key strategies for *exact* inference:
    - **Belief propagation**, for inference on *acyclic* graphs
    - **Junction tree algorithm**, for making *any* graph acyclic
      (by merging variables and factors: blows up the runtime)

Factor Graph Notation

- Variables:

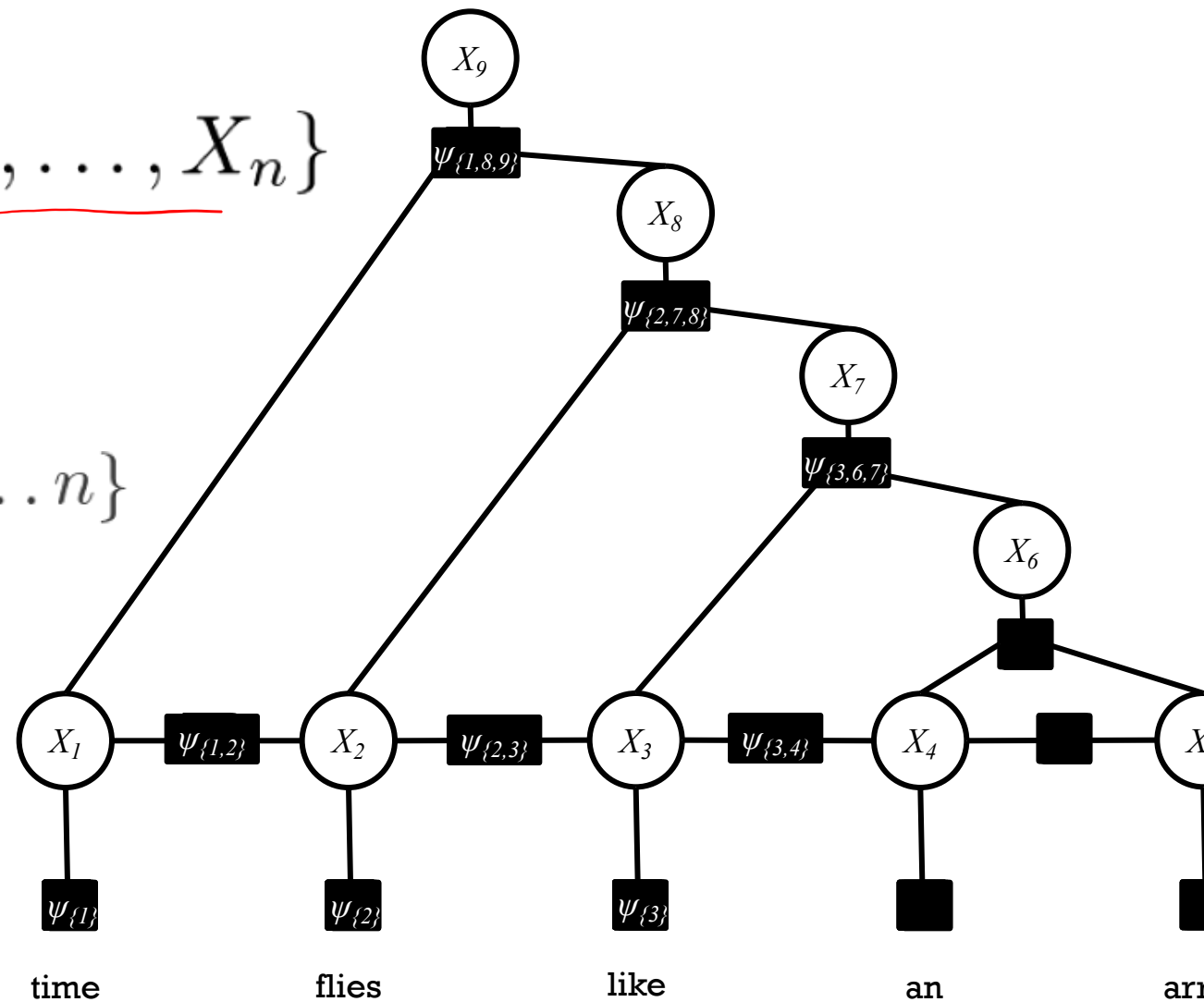$$\mathcal{X} = \{X_1, \ldots, X_i, \ldots, X_n\}$$

- Factors:

$$\psi_\alpha, \psi_\beta, \psi_\gamma, \ldots$$

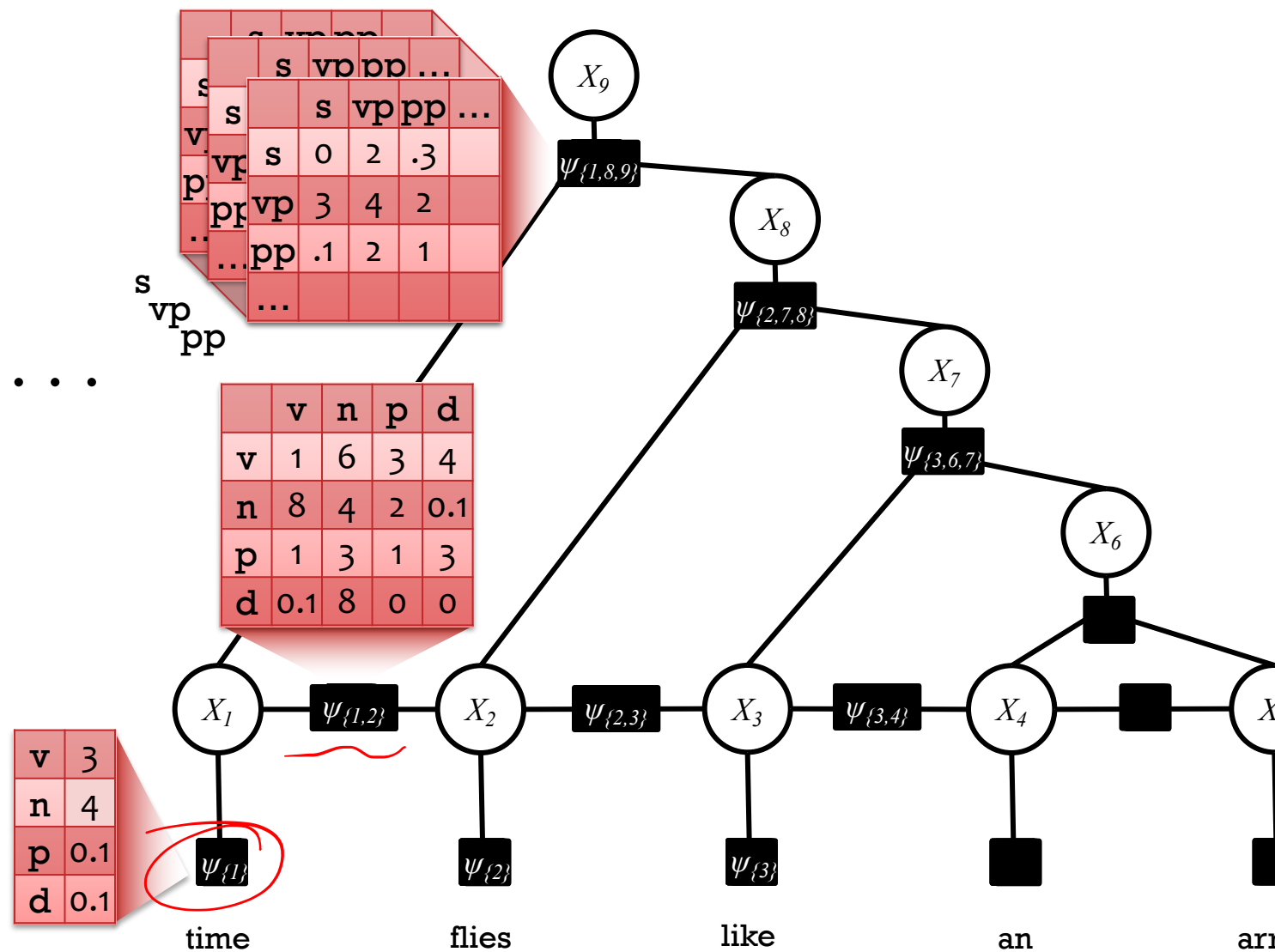where $\alpha, \beta, \gamma, \ldots \subseteq \{1, \ldots n\}$

Joint Distribution

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_\alpha \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}})$$

Factors are Tensors

- Factors:

$$\psi_\alpha, \psi_\beta, \psi_\gamma, \ldots$$

An Inference Example

$m_e(d)$

$$p(a,b) = f_1(a,b) \sum_{c,d} f_2(b,c,d) f_3(c) f_5(d) \sum_e f_4(d,e)$$

$\underbrace{}_{\mu_{f_5 \to d}(d)}$ $\underbrace{}_{\mu_{f_4 \to d}(d)}$

$f_1(a,b) = p(a|b)$

$p(a|b)p(b|c,d)p(c)p(d)p(e|d)$

$f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d)$

$\underbrace{}_{\mu_{c \to f_2}(c)}$ $\underbrace{}_{\mu_{d \to f_2}(d)}$

$\underbrace{}_{\mu_{f_2 \to b}(b)}$

$$p(a) = \sum_b f_1(a,b) \mu_{f_2 \to b}(b)$$

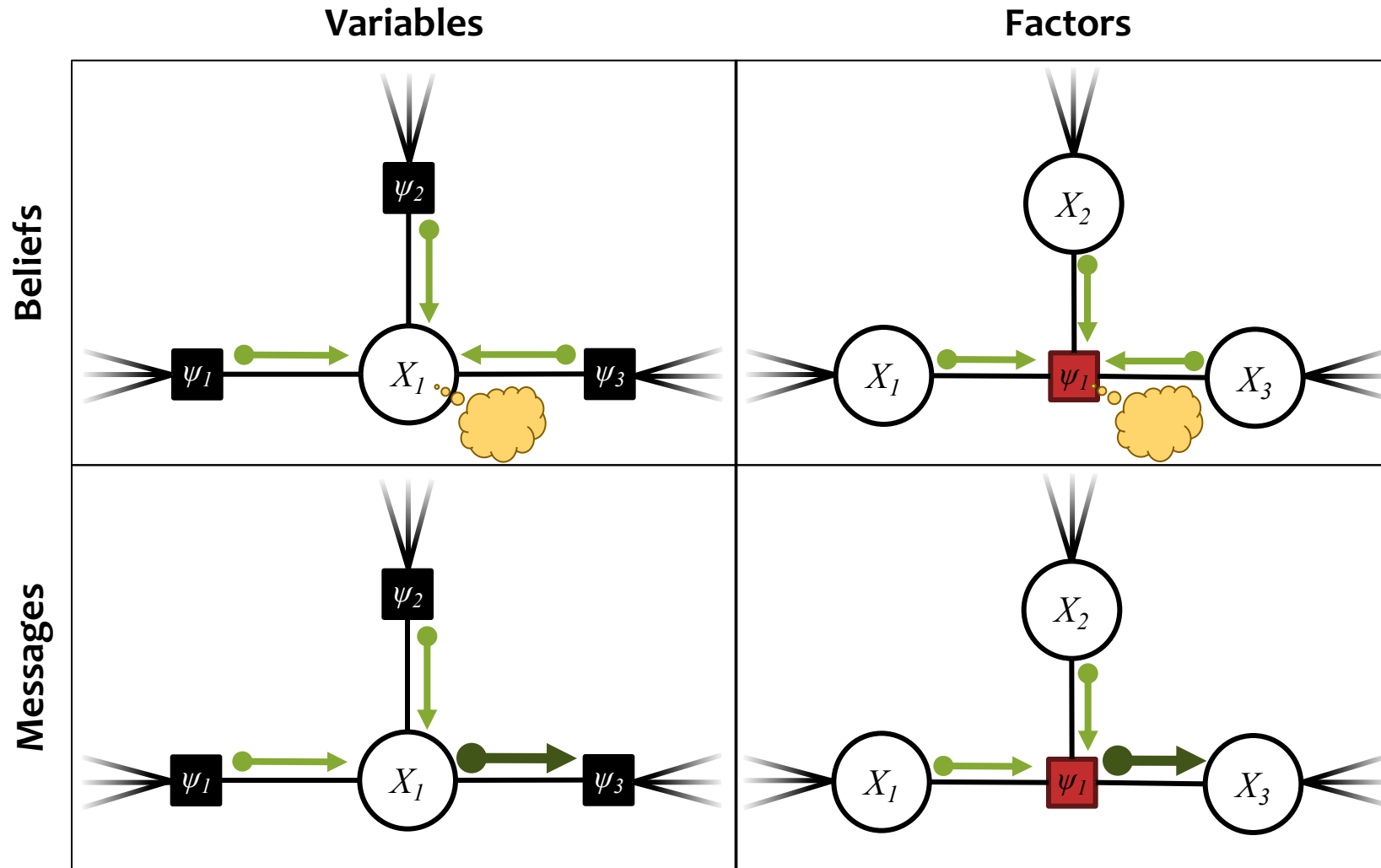$\underbrace{}_{\mu_{f_1 \to a}(a)}$

15
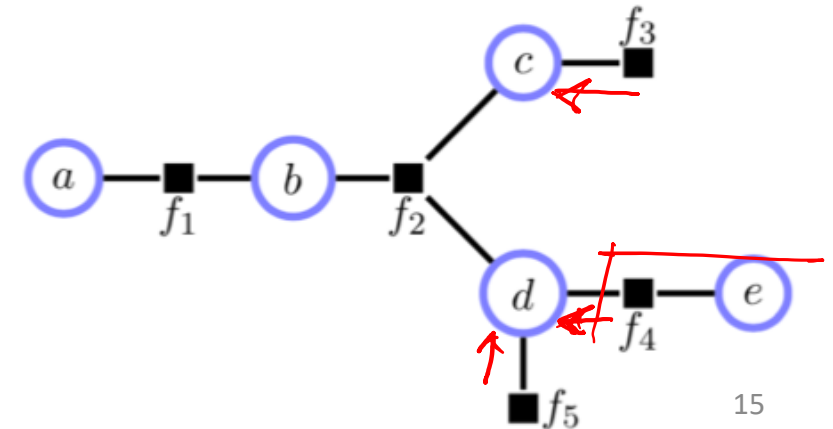
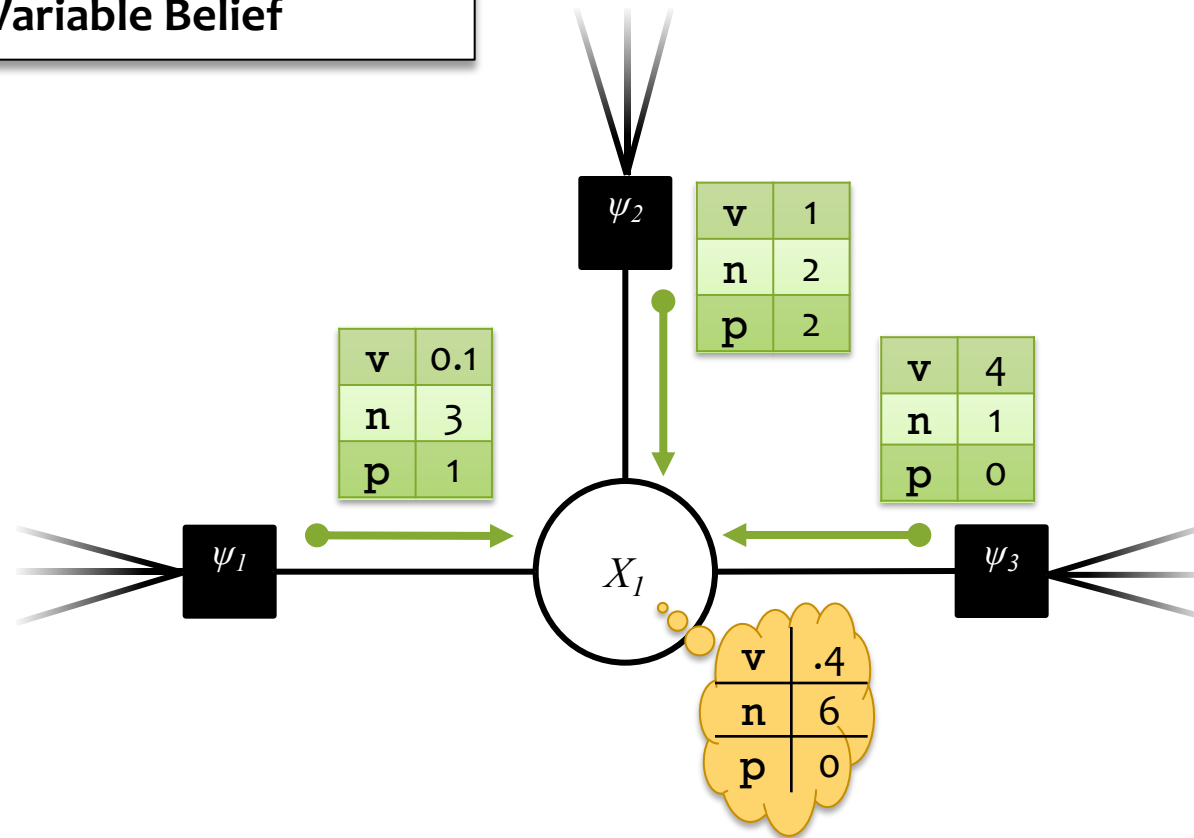# Message Passing in Belief Propagation
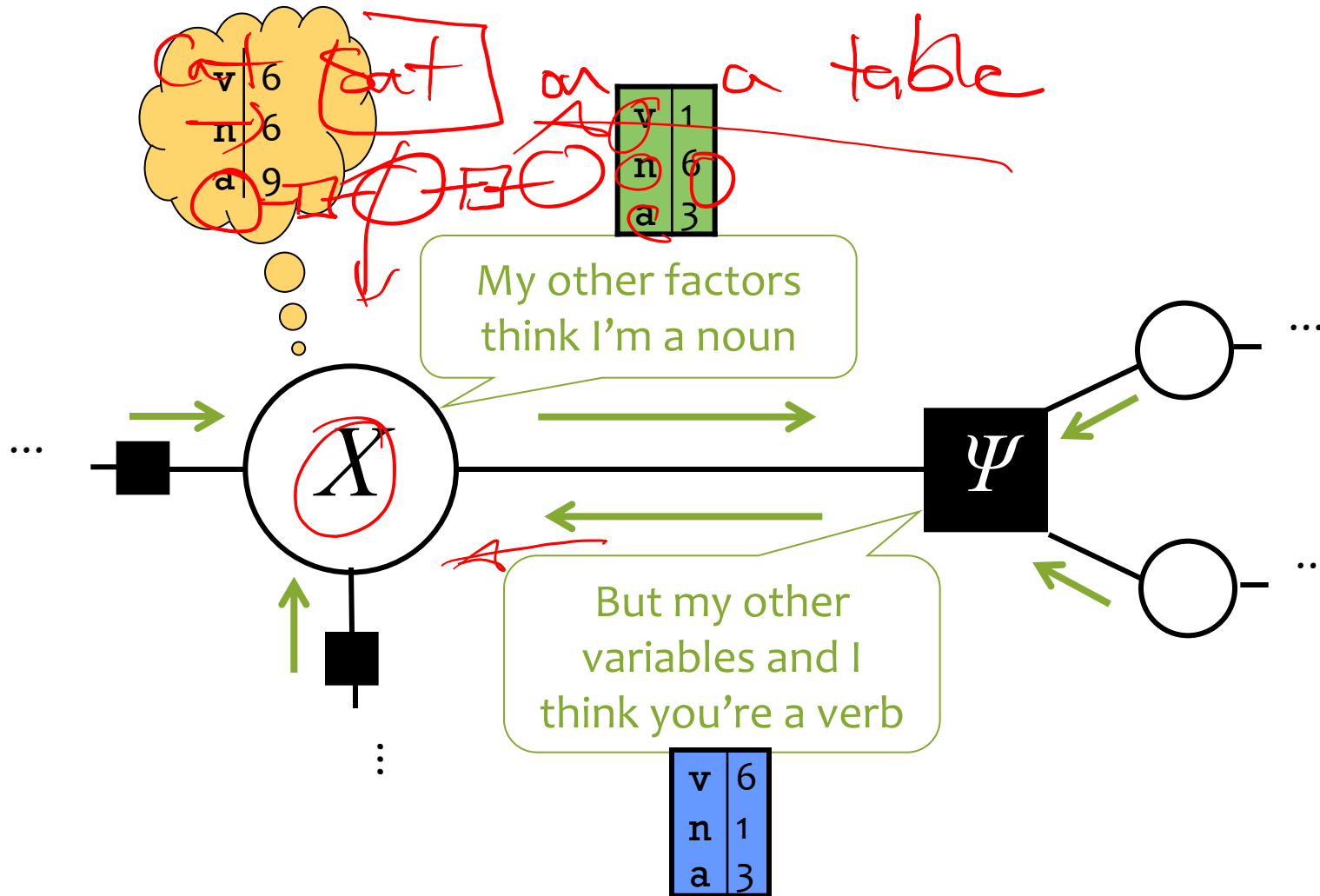
Both of these messages judge the possible values of variable X.
Their product = belief at X = product of all 3 messages to X.

Sum-Product Belief Propagation

Variables

Factors

Beliefs

Messages

Sum-Product Belief Propagation

Variable Belief

v	1
n	2
p	2

v	0.1
n	3
p	1

v	4
n	1
p	0

v	.4
n	6
p	0

$$b_i(x_i) = \prod_{\alpha \in \mathcal{N}(i)} \mu_{\alpha \to i}(x_i)$$

Sum-Product Belief Propagation

$$\mu_{i \to \alpha}(x_i) = \prod_{\alpha \in \mathcal{N}(i) \setminus \alpha} \mu_{\alpha \to i}(x_i)$$

Sum-Product Belief Propagation

Factor Belief

	v	n
p	0.1	8
d	3	0
n	1	1

p	4
d	1
n	0

v	8
n	0.2

X_1 ψ_1 X_3

	v	n
p	3.2	6.4
d	24	0
n	0	0

$$b_\alpha(\boldsymbol{x_\alpha}) = \psi_\alpha(\boldsymbol{x_\alpha}) \prod_{i \in \mathcal{N}(\alpha)} \mu_{i \to \alpha}(\boldsymbol{x_\alpha}[i])$$

Sum-Product Belief Propagation

Factor Belief

$$b_\alpha(\boldsymbol{x_\alpha}) = \psi_\alpha(\boldsymbol{x_\alpha}) \prod_{i \in \mathcal{N}(\alpha)} \mu_{i \to \alpha}(\boldsymbol{x_\alpha}[i])$$

Slides adapted from
Matt Gormley (2016)

Sum-Product Belief Propagation

Factor Message
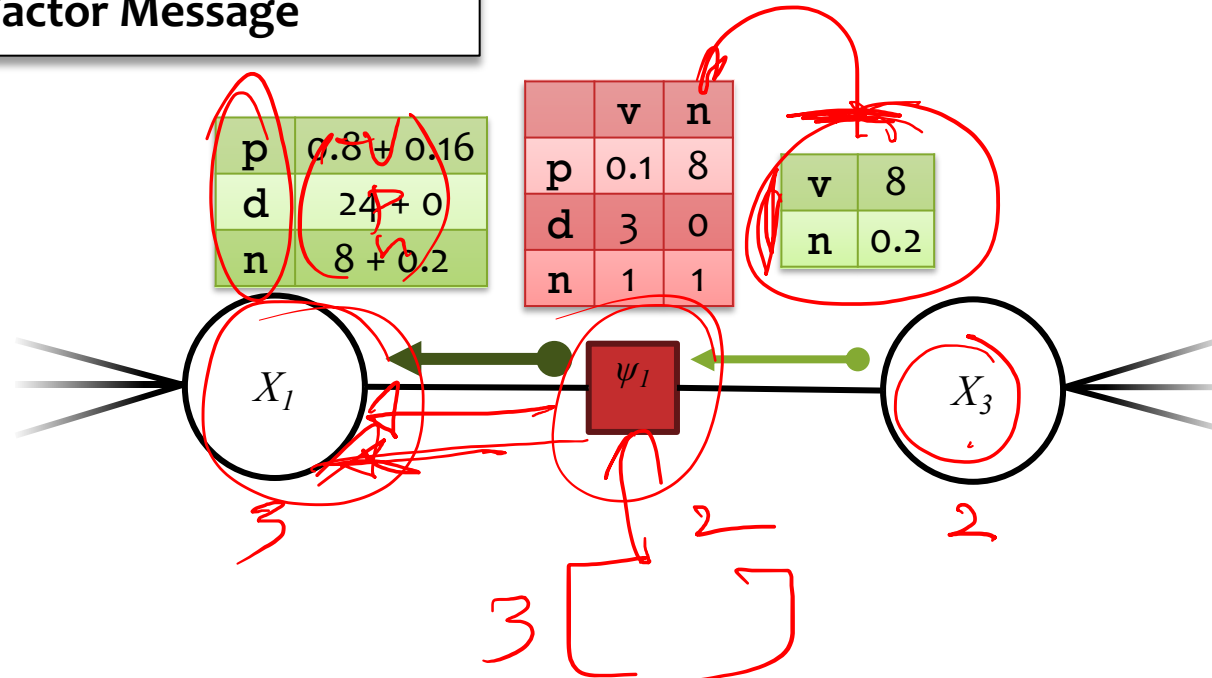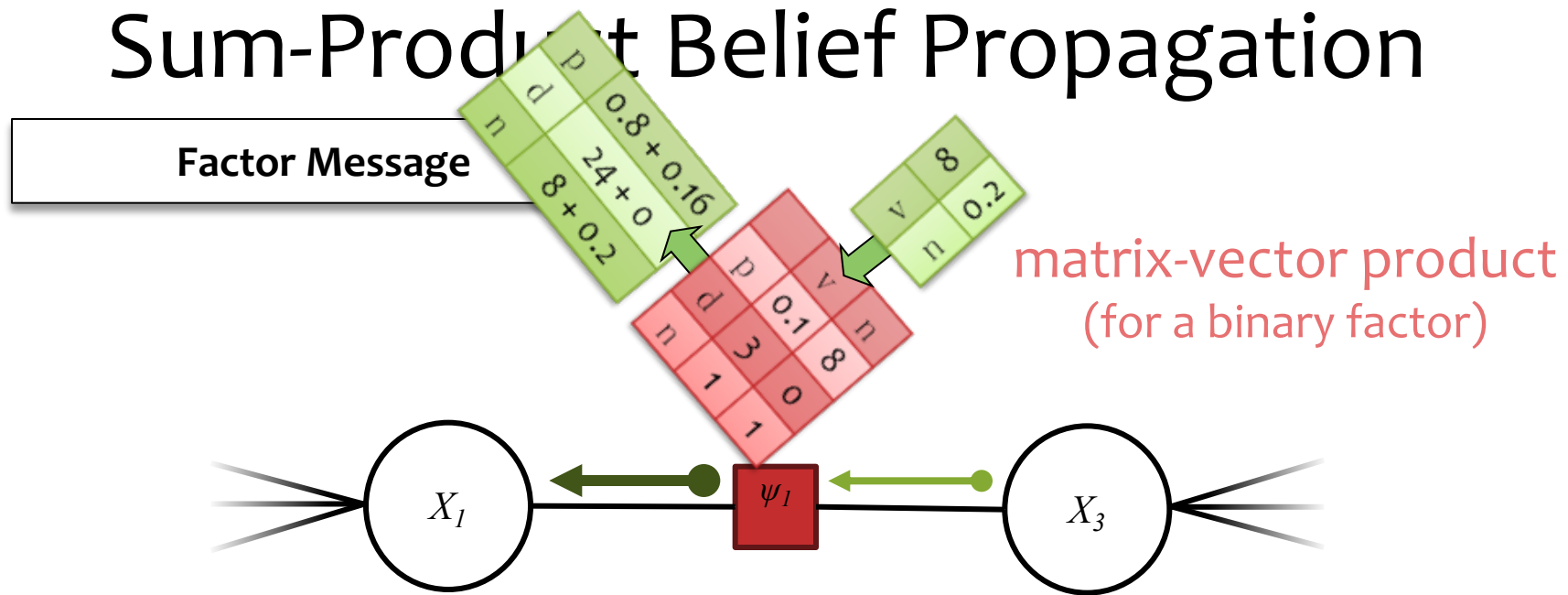

	v	n
p	0.1	8
d	3	0
n	1	1

p	0.8 + 0.16
d	24 + 0
n	8 + 0.2

v	8
n	0.2

$$\mu_{\alpha \to i}(x_i) = \sum_{\boldsymbol{x_\alpha} : \boldsymbol{x_\alpha}[i] = x_i} \psi_\alpha(\boldsymbol{x_\alpha}) \prod_{j \in \mathcal{N}(\alpha) \setminus i} \mu_{j \to \alpha}(\boldsymbol{x_\alpha}[i])$$

$$\sum_{x_3} \psi_1(x_3, \cdot) \mu_{X_3 \to \psi_1}$$
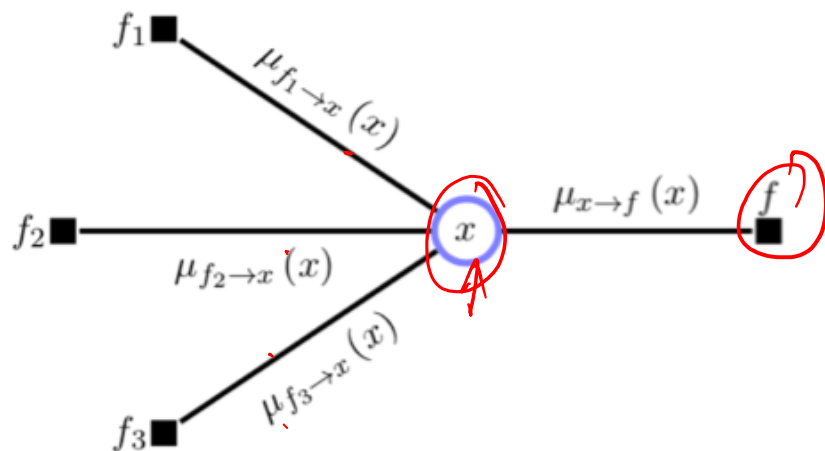
Sum-Product Belief Propagation

Factor Message

matrix-vector product
(for a binary factor)

X_1 ψ_1 X_3

$$\mu_{\alpha \to i}(x_i) = \sum_{\boldsymbol{x_\alpha} : \boldsymbol{x_\alpha}[i] = x_i} \psi_\alpha(\boldsymbol{x_\alpha}) \prod_{j \in \mathcal{N}(\alpha) \backslash i} \mu_{j \to \alpha}(\boldsymbol{x_\alpha}[i])$$

Variable to Factor message

$$\mu_{x \to f}(x) = \prod_{g \in \{\text{ne}(x) \setminus f\}} \mu_{g \to x}(x)$$

Factor to Variable message

$$\mu_{f \to x}(x) = \max_{\mathcal{X}_f \setminus x} \phi_f(\mathcal{X}_f) \prod_{y \in \{\text{ne}(f) \setminus x\}} \mu_{y \to f}(y)$$

Marginal

$$p(x) \propto \prod_{f \in \text{ne}(x)} \mu_{f \to x}(x)$$
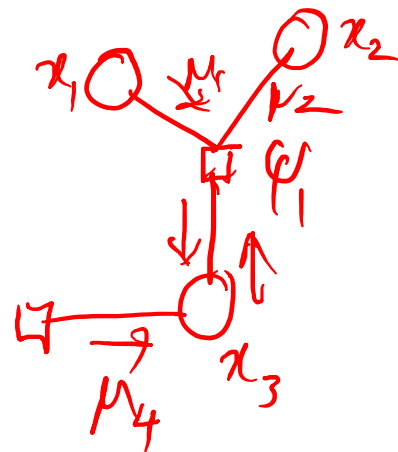
Sum-Product Belief Propagation

Input: a factor graph with no cycles
Output: exact marginals for each variable and factor

Algorithm:
1. Initialize the messages to the uniform distribution.

$$\mu_{i \to \alpha}(x_i) = 1 \qquad \mu_{\alpha \to i}(x_i) = 1$$

1. Choose a root node.
2. Send messages from the **leaves** to the **root**.
   Send messages from the **root** to the **leaves**.

$$\mu_{i \to \alpha}(x_i) = \prod_{\alpha \in \mathcal{N}(i) \backslash \alpha} \mu_{\alpha \to i}(x_i) \qquad \mu_{\alpha \to i}(x_i) = \sum_{\boldsymbol{x_\alpha} : \boldsymbol{x_\alpha}[i] = x_i} \psi_\alpha(\boldsymbol{x_\alpha}) \prod_{j \in \mathcal{N}(\alpha) \backslash i} \mu_{j \to \alpha}(\boldsymbol{x_\alpha}[i])$$
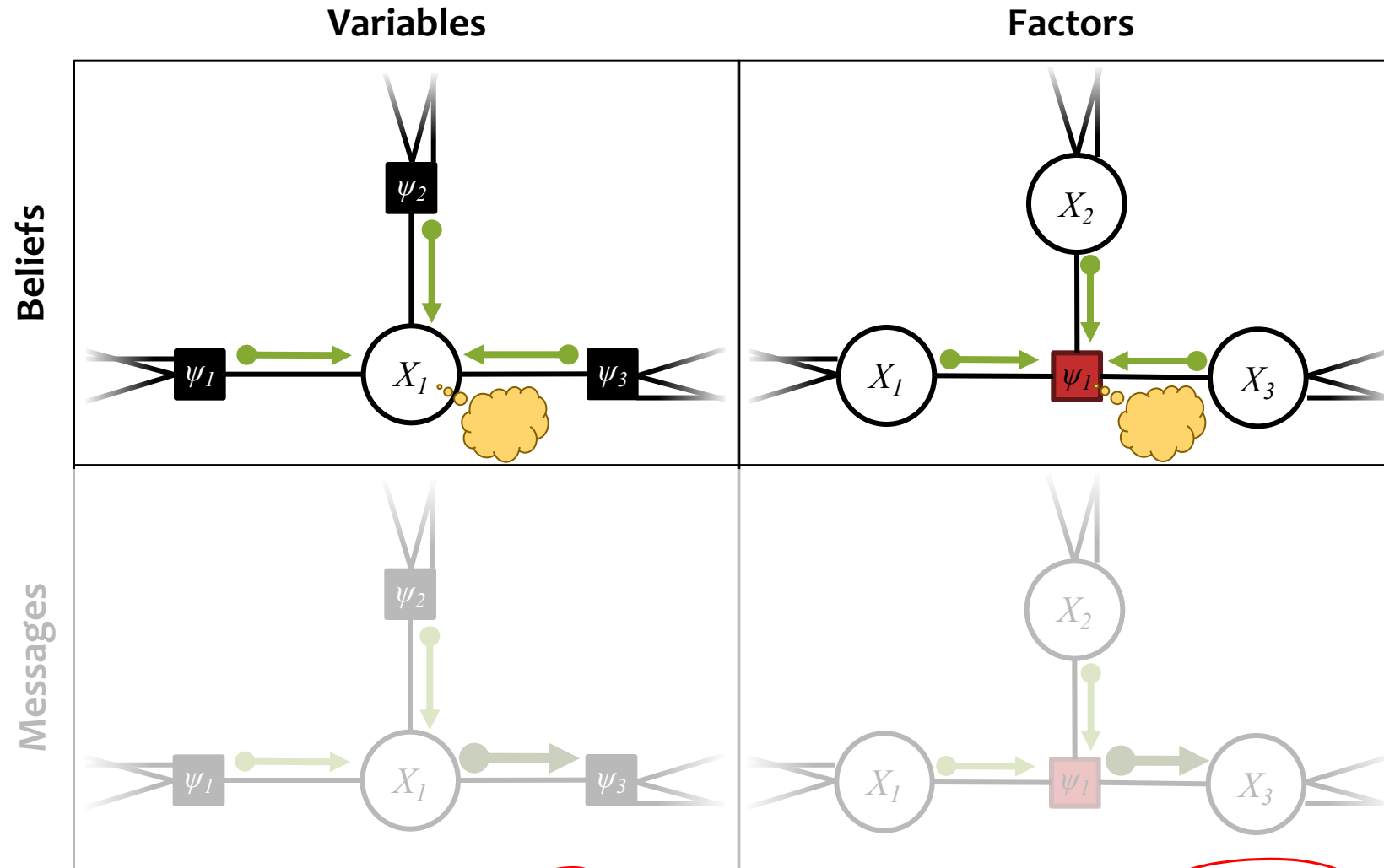
1. Compute the beliefs (unnormalized marginals).

$$b_i(x_i) = \prod_{\alpha \in \mathcal{N}(i)} \mu_{\alpha \to i}(x_i) \qquad b_\alpha(\boldsymbol{x_\alpha}) = \psi_\alpha(\boldsymbol{x_\alpha}) \prod_{i \in \mathcal{N}(\alpha)} \mu_{i \to \alpha}(\boldsymbol{x_\alpha}[i])$$

2. Normalize beliefs and return the **exact** marginals.

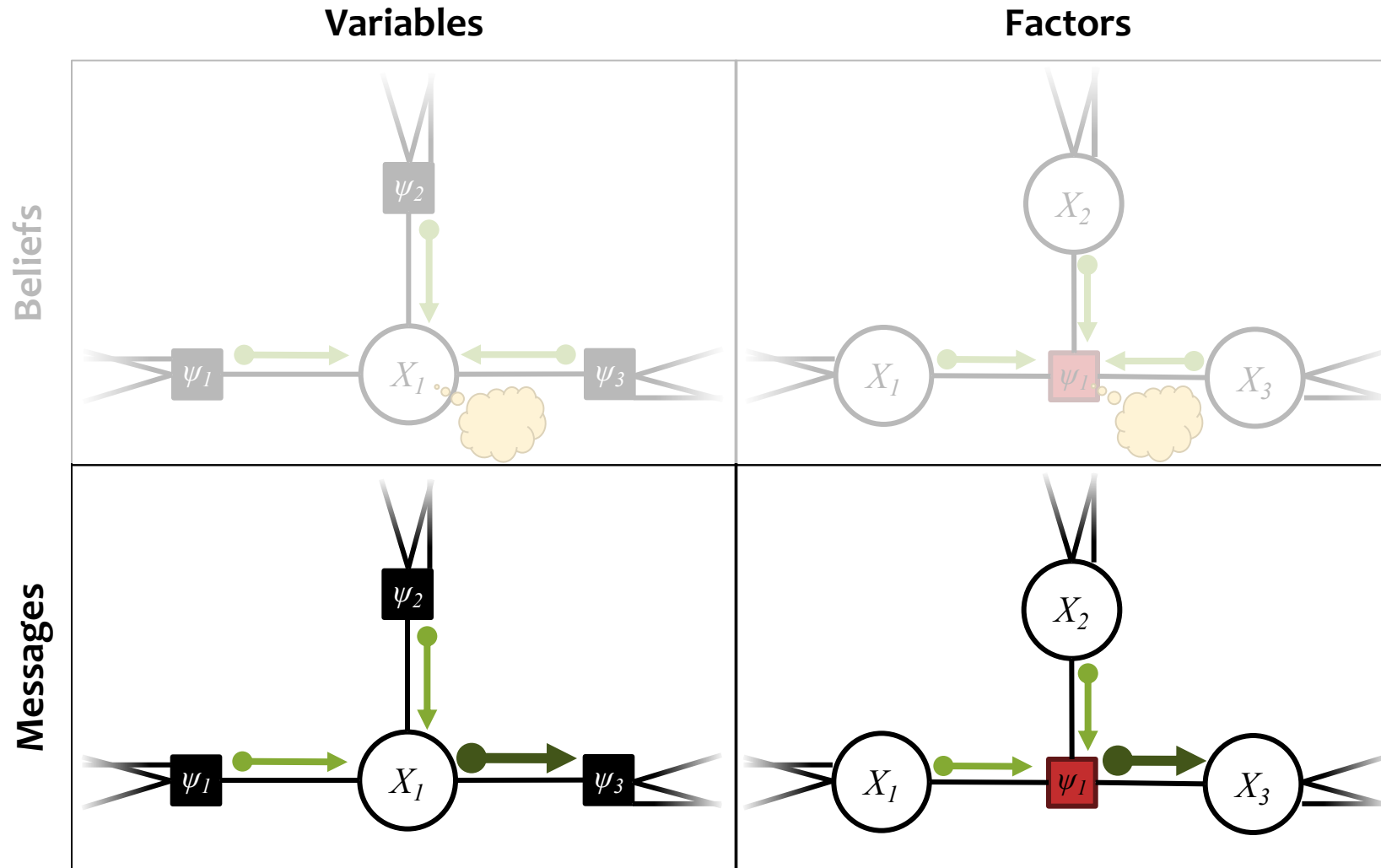$$p_i(x_i) \propto b_i(x_i) \qquad p_\alpha(\boldsymbol{x_\alpha}) \propto b_\alpha(\boldsymbol{x_\alpha})$$

Sum-Product Belief Propagation

Variables

Factors

Beliefs

$$b_i(x_i) = \prod_{\alpha \in \mathcal{N}(i)} \mu_{\alpha \to i}(x_i)$$

$$b_\alpha(\boldsymbol{x_\alpha}) = \psi_\alpha(\boldsymbol{x_\alpha}) \prod_{i \in \mathcal{N}(\alpha)} \mu_{i \to \alpha}(\boldsymbol{x_\alpha}[i])$$

26

Sum-Product Belief Propagation

Variables

Factors

Beliefs

Messages

$$\mu_{i \to \alpha}(x_i) = \prod_{\alpha \in \mathcal{N}(i) \setminus \alpha} \mu_{\alpha \to i}(x_i)$$

$$\mu_{\alpha \to i}(x_i) = \sum_{\boldsymbol{x_\alpha} : \boldsymbol{x_\alpha}[i] = x_i} \psi_\alpha(\boldsymbol{x_\alpha}) \prod_{j \in \mathcal{N}(\alpha) \setminus i} \mu_{j \to \alpha}(\boldsymbol{x_\alpha}[i])$$

(Acyclic) Belief Propagation

In a factor graph with no cycles:

1. Pick any node to serve as the root.
2. Send messages from the **leaves** to the **root**.
3. Send messages from the **root** to the **leaves**.

A node computes an outgoing message along an edge
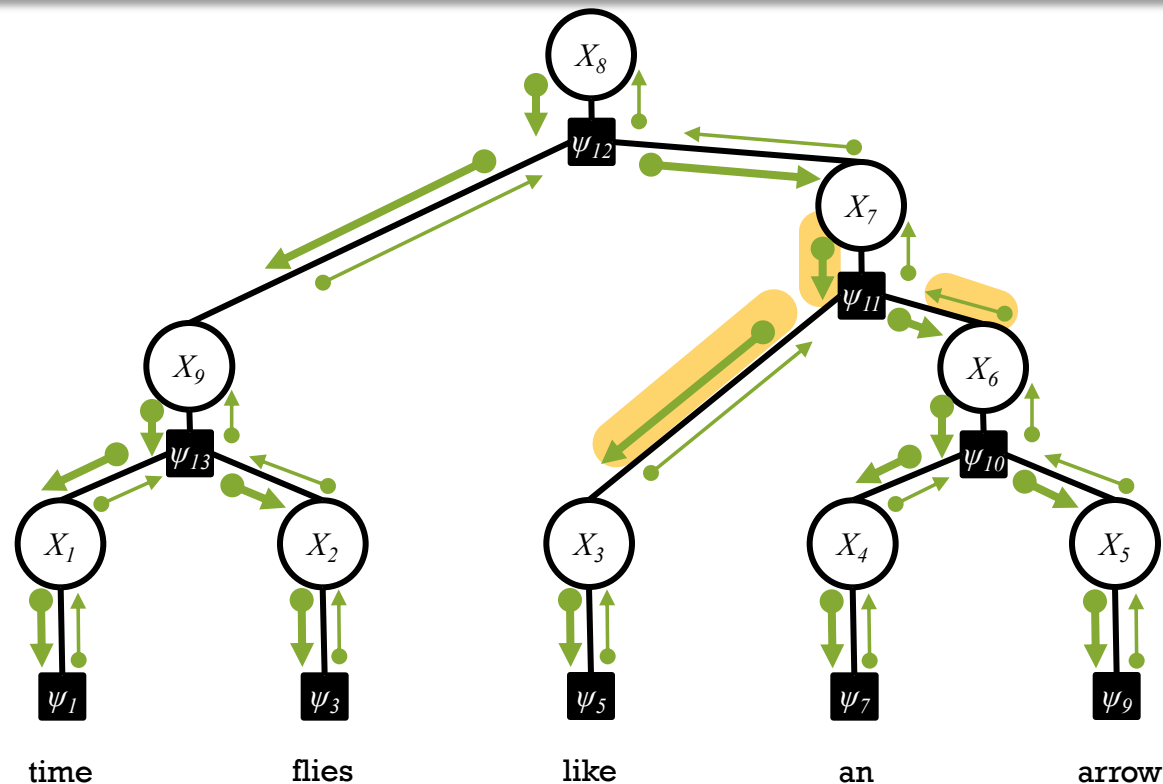   only after it has received incoming messages along all its other edges.

28

(Acyclic) Belief Propagation

In a factor graph with no cycles:

1. Pick any node to serve as the root.
2. Send messages from the **leaves** to the **root**.
3. Send messages from the **root** to the **leaves**.

A node computes an outgoing message along an edge
   only after it has received incoming messages along all its other edges.

A note on the implementation

To avoid numerical precision issue, use log message ($\lambda = \log \mu$):
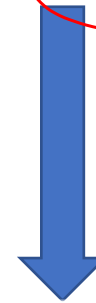
Variable to Factor message

$$\mu_{x \to f}(x) = \prod_{g \in \{\mathrm{ne}(x) \backslash f\}} \mu_{g \to x}(x)$$

$$\lambda_{x \to f}(x) = \sum_{g \in \{\mathrm{ne}(x) \backslash f\}} \lambda_{g \to x}(x)$$

Factor to Variable message

$$\mu_{f \to x}(x) = \max_{\mathcal{X}_f \backslash x} \phi_f(\mathcal{X}_f) \prod_{y \in \{\mathrm{ne}(f) \backslash x\}} \mu_{y \to f}(y)$$

$$\lambda_{f \to x}(x) = \log \left(\sum_{\mathcal{X}_f \backslash x} \phi_f(\mathcal{X}_f) \exp \left(\sum_{y \in \{\mathrm{ne}(f) \backslash x\}} \lambda_{y \to f}(y) \right) \right)$$

How about other queries?
(MPA, Evidence)

Example
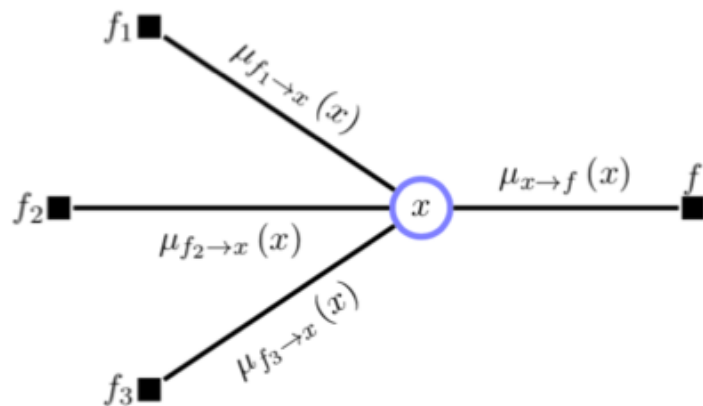
$$\max_{\mathbf{x}} f(\mathbf{x}) = \max_{x_1,x_2,x_3,x_4} \phi(x_1,x_2)\phi(x_2,x_3)\phi(x_3,x_4) = \max_{x_1,x_2,x_3} \phi(x_1,x_2)\phi(x_2,x_3)\underbrace{\max_{x_4} \phi(x_3,x_4)}_{\gamma_4(x_3)}$$

$m_{x_3}(x_3)$

$$\underbrace{\max_{x_3} \phi(x_2,x_3)\gamma_4(x_3)}_{\gamma_3(x_2)}$$

$$\max_{x_1} \underbrace{\max_{x_2} \phi(x_1,x_2)\gamma_3(x_2)}_{\gamma_2(x_1)}$$

The Max Product Algorithm

$P(x_2)$?

$\max_{x_1} P(\mathcal{X})$

$\psi_1(x_1, x_3) \psi(x, x_2)$

Variable to Factor message

$$\mu_{x \to f}(x) = \prod_{g \in \{\mathrm{ne}(x) \backslash f\}} \mu_{g \to x}(x)$$

Factor to Variable message

$$\mu_{f \to x}(x) = \max_{\mathcal{X}_f \backslash x} \phi_f(\mathcal{X}_f) \prod_{y \in \{\mathrm{ne}(f) \backslash x\}} \mu_{y \to f}(y)$$

$P(x_1, x_2, x_3, x_4, x_5)$

ψ_4 ψ_5 x_2 ψ_3 x_3 ψ_1 x_4 ψ_2 x_5

$\max_{x_3, x_1} P(x_5, x_1) P(x_4) P(x_3)$

Maximal State

$$x^* = \underset{x}{\mathrm{argmax}} \prod_{f \in \mathrm{ne}(x)} \mu_{f \to x}(x)$$

$P(x_1) = \sum_{x_2, x_3, x_4, x_5}$

33
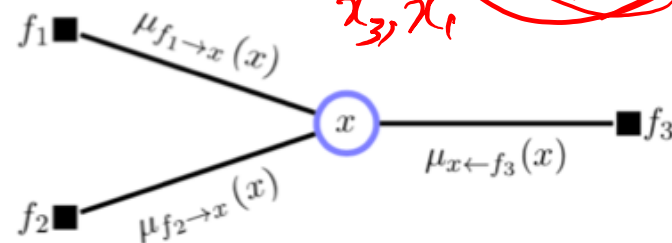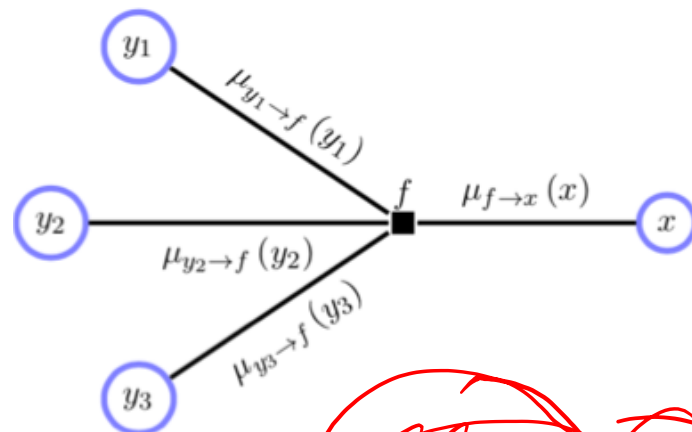
Can I use BP in a multiply connected graph?

Loops the trouble makers

$$p(a, b, c, d) = f_1(a, b) \, f_2(b, c) \, f_3(c, d) \, f_4(a, d)$$
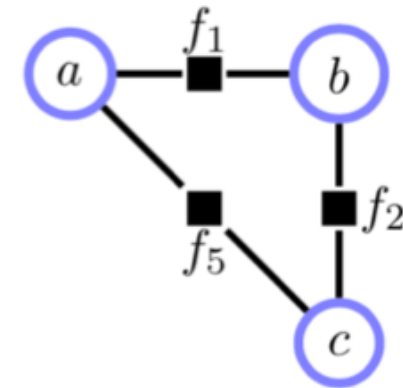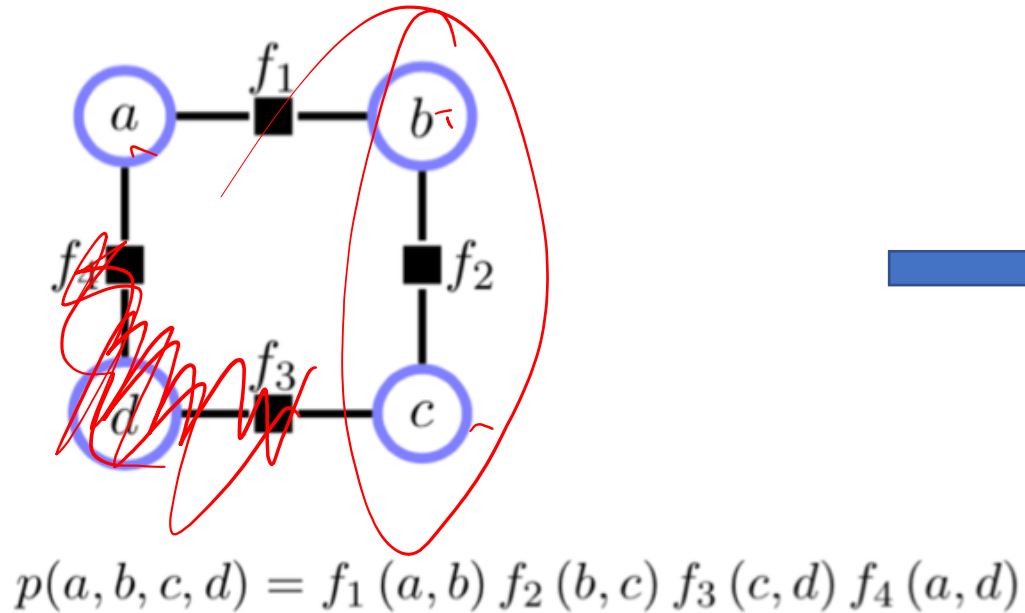
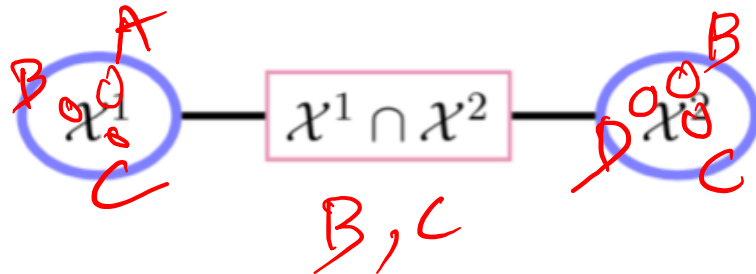$$p(a, b, c) = f_1(a, b) \, f_2(b, c) \underbrace{\sum_d f_3(c, d) \, f_4(a, d)}_{f_5(a,c)}$$

- One needs to account for the fact that the structure of the graph changes.
- The junction tree algorithm deals with this by combining variables to make a new singly connected graph for which the graph structure remains singly connected under variable elimination.

Clique Graph

- **Def (Clique Graph)**: A clique graph consists of a set of potentials, $\phi_1(\chi_1), \cdots, \phi_n(\chi_n)$ each defined on a set of variables χ_1. For neighboring cliques on the graph, defined on sets of variables χ_1 and χ_j, the intersection $\chi_s = \chi_i \cap \chi_j$ is called the separator and has a corresponding potential $\phi_s(\chi_s)$. A clique graph represents the <u style="color:red">function</u>

$$\frac{\prod_c \phi_c(\mathcal{X}^c)}{\prod_s \phi_s(\mathcal{X}^s)}$$

Example

$$\phi(\mathcal{X}^1)\phi(\mathcal{X}^2)/\phi(\mathcal{X}^1 \cap \mathcal{X}^2)$$

$$\frac{\phi(A,B,C)\,\phi(B,C,D)}{\phi(B,C)}$$

Clique Graph

- **Def (Clique Graph)**: A clique graph consists of a set of potentials, $\phi_1(\chi_1), \cdots, \phi_n(\chi_n)$ each defined on a set of variables χ_1. For neighboring cliques on the graph, defined on sets of variables χ_1 and χ_j, the intersection $\chi_s = \chi_i \cap \chi_j$ is called the separator and has a corresponding potential $\phi_s(\chi_s)$. A clique graph represents the function

$$\frac{\prod_c \phi_c(\mathcal{X}^c)}{\prod_s \phi_s(\chi_s)}$$

Don't confuse it with Factor Graph!

Example

$$\mathcal{X}^1 - \boxed{\mathcal{X}^1 \cap \mathcal{X}^2} - \mathcal{X}^2 \qquad \phi(\mathcal{X}^1)\phi(\mathcal{X}^2)/\phi(\mathcal{X}^1 \cap \mathcal{X}^2)$$

Example: probability density

$$Z \, p(a,b,c) = \sum_d \phi(a,b,c)\phi(b,c,d)$$

$$Z \, p(b,c,d) = \sum_a \phi(a,b,c)\phi(b,c,d)$$
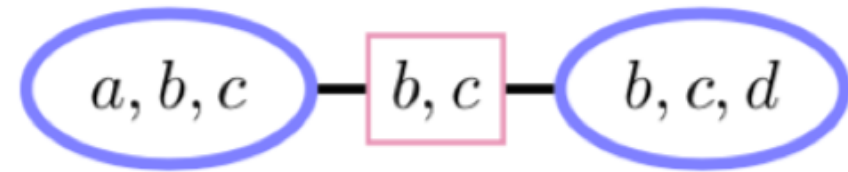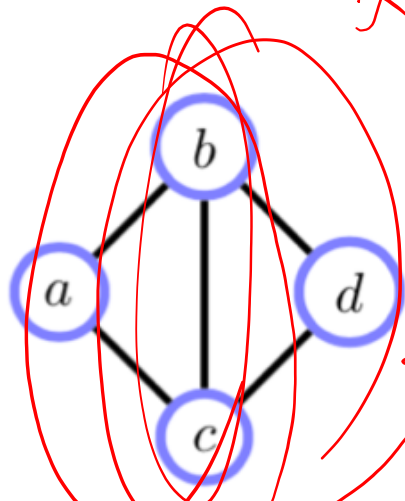
$$\sum_{a,d} \phi(a,b,c)$$

$$p(a,b,c,d) = \frac{\phi(a,b,c)\phi(b,c,d)}{Z} = \frac{p(a,b,c)p(b,c,d)}{p(c,b)}$$

$$p(a,b,c,d) = \frac{\phi(a,b,c)\,\phi(b,c,d)}{\psi(bc)}$$

Junction Tree

- **Idea**: form a new representation of the graph in which variables are clustered together, resulting in a singly-connected graph in the cluster variables.

- **Insight**: distribution can be written as product of marginal distributions, divided by a product of the intersection of the marginal distributions.

- Not a remedy to the intractability.

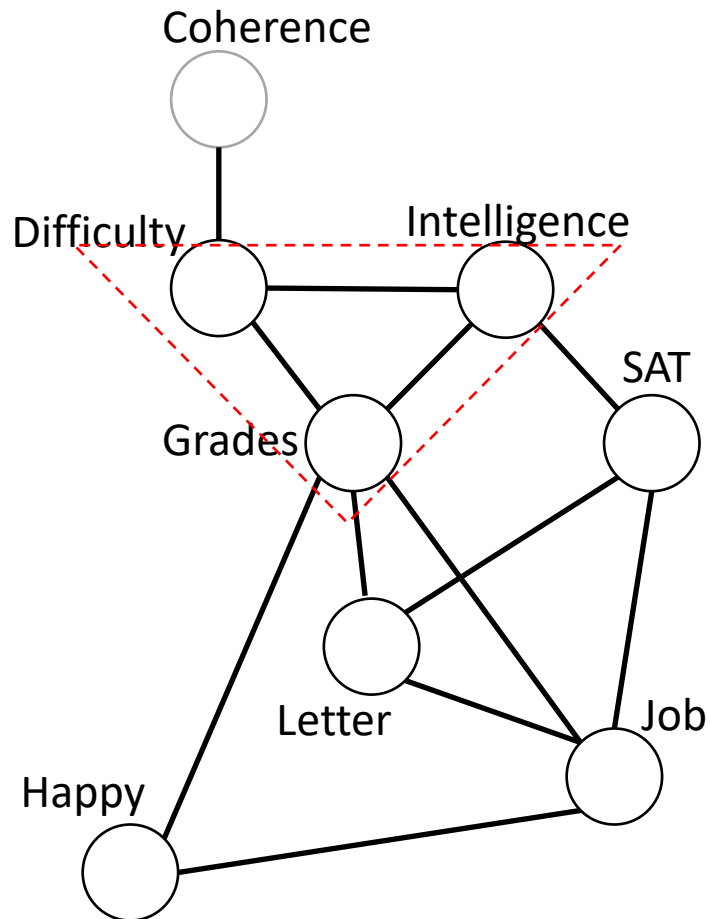# Let's learn by an example….
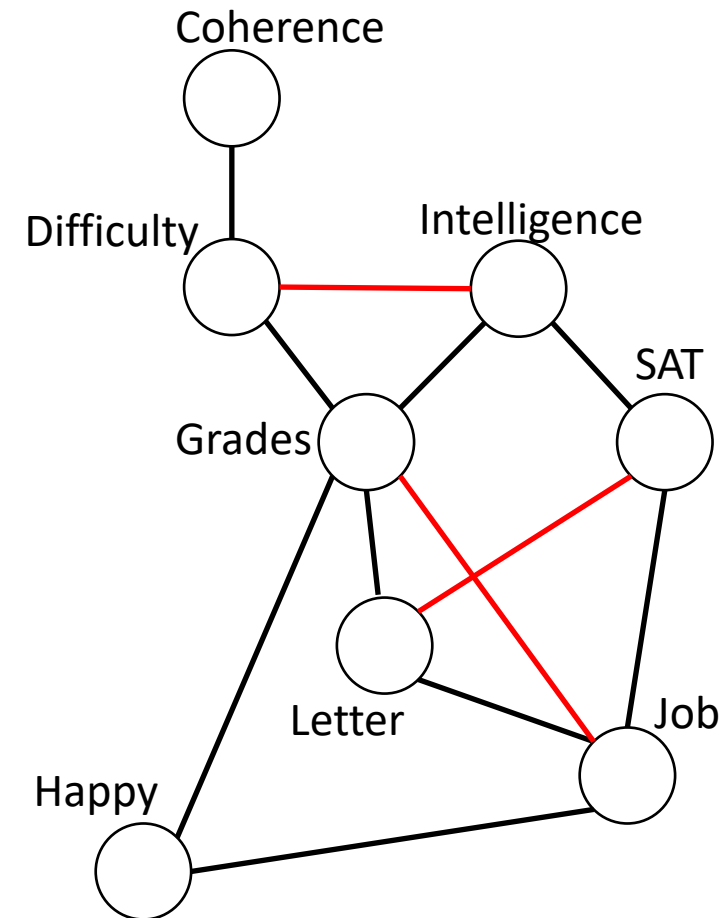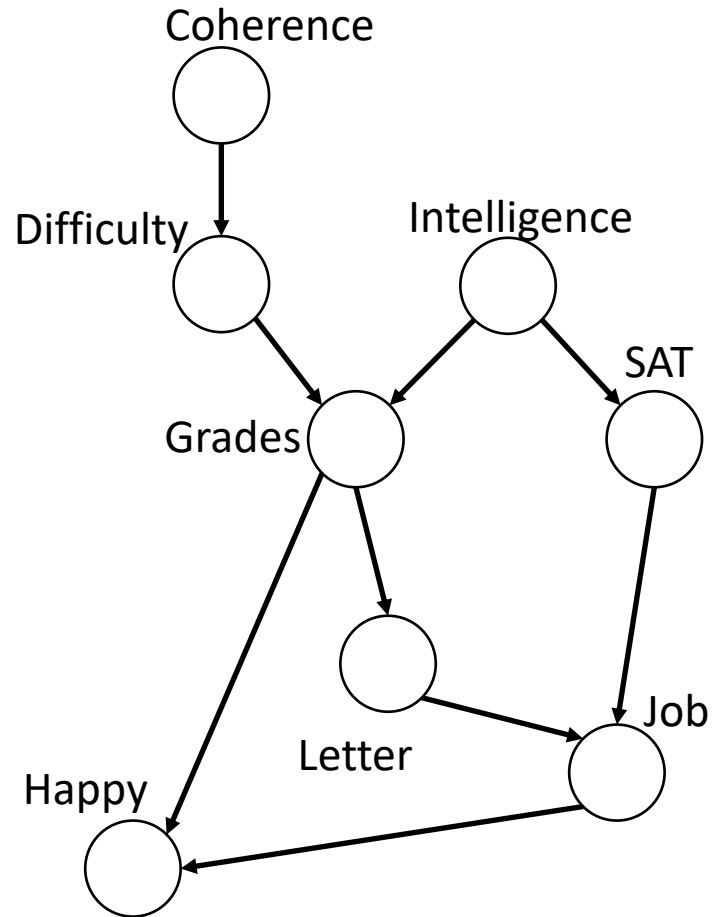
Moralization

Let's learn by an example....
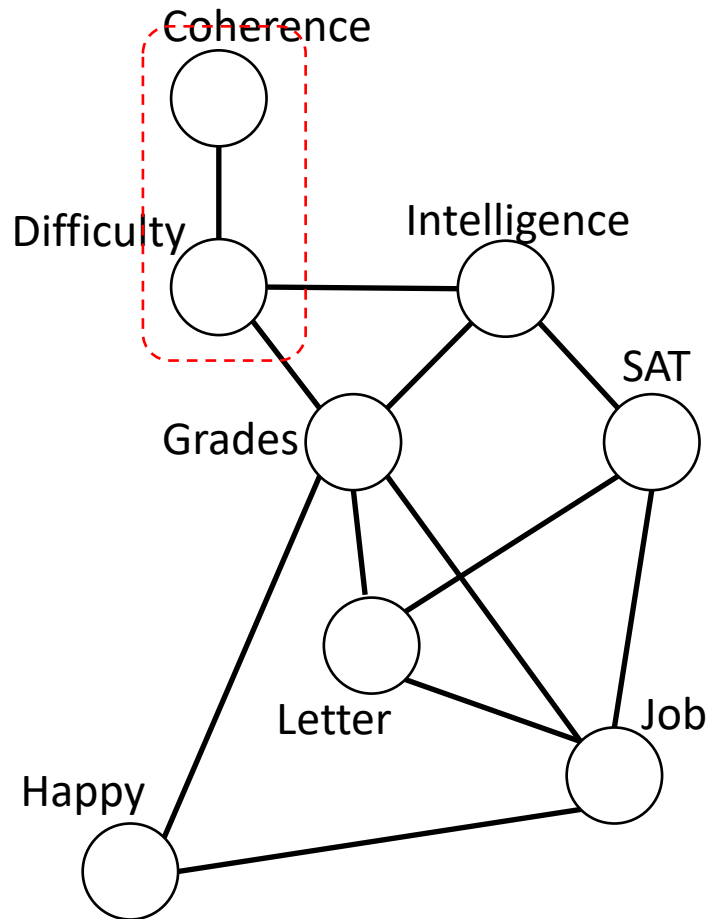
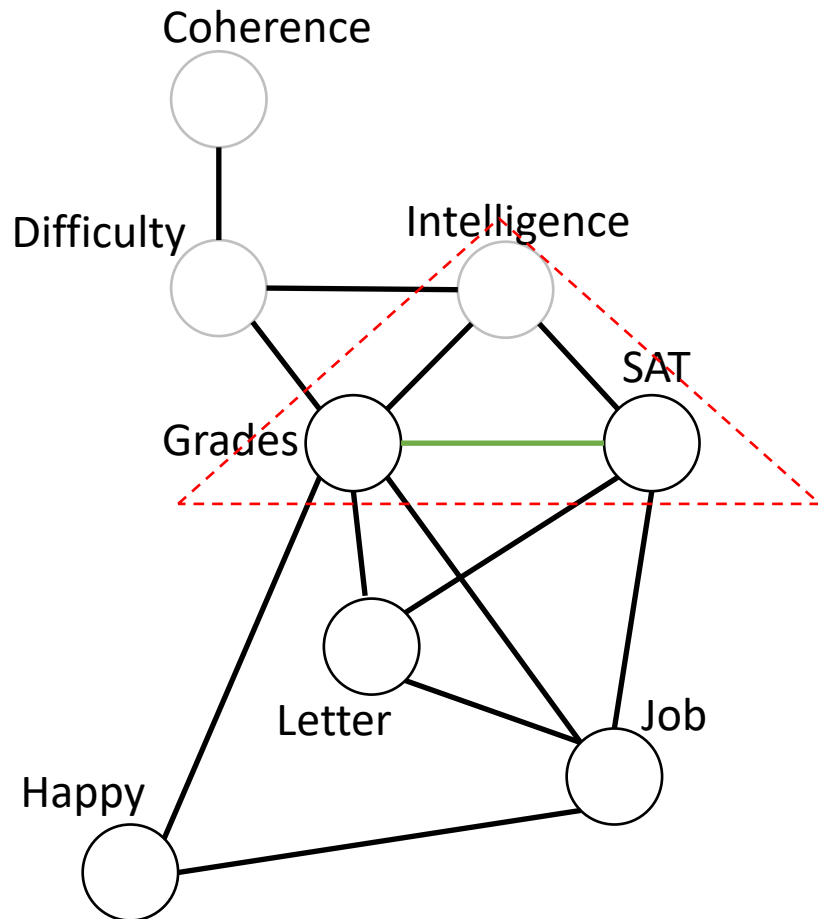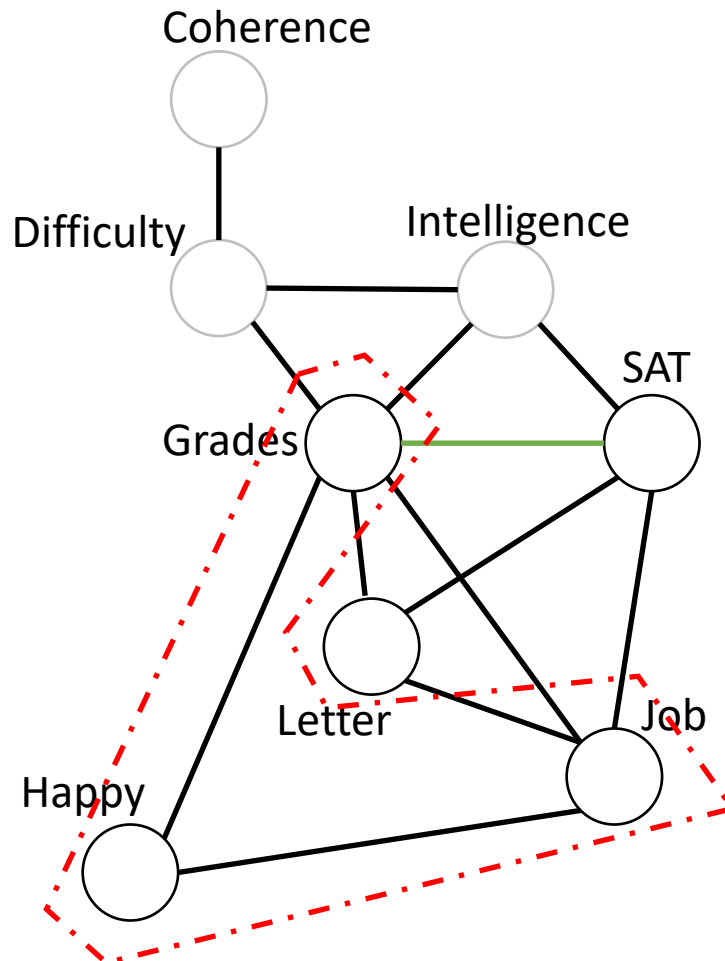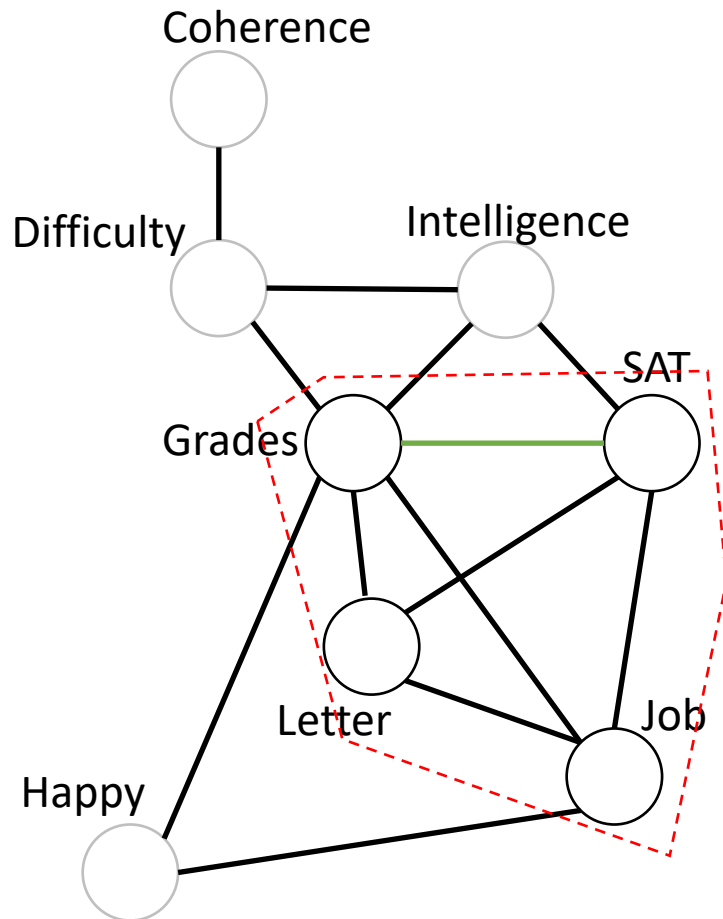Let's pick an ordering for the variable elimination
C, D, I, H, G, S, L

Let's learn by an example....

Let's pick an ordering for the variable elimination

C, D, I, H, G, S, L

Let's learn by an example….

Let's pick an ordering for the variable elimination

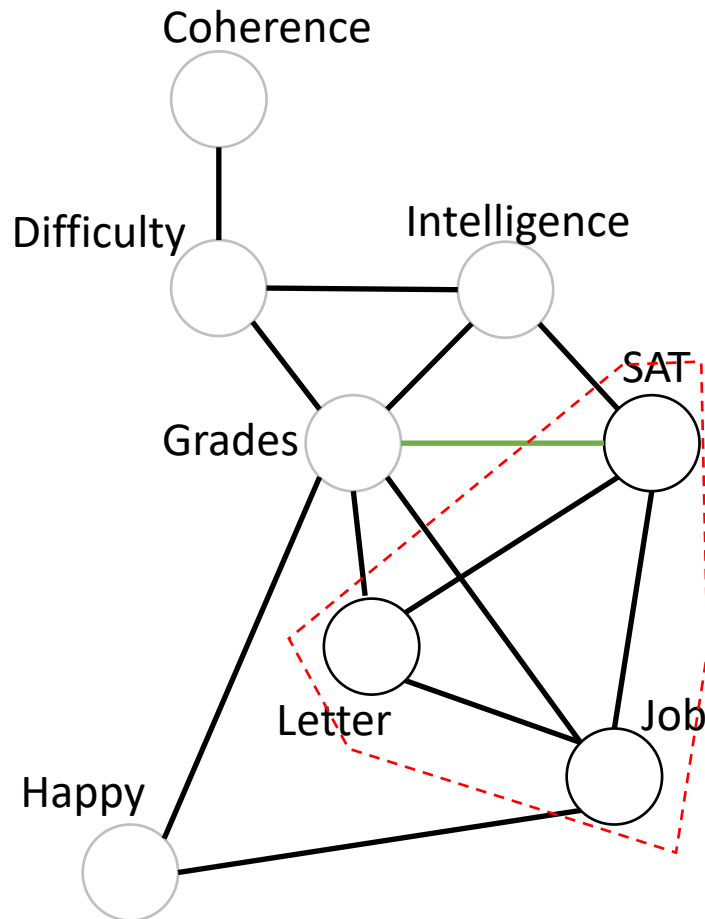C, D, I, H, G, S, L

Let's learn by an example….

Let's pick an ordering for the variable elimination
C, D, I, H, G, S, L

Let's learn by an example....

Let's pick an ordering for the variable elimination

C, D, I, H, G, S, L

Let's learn by an example....

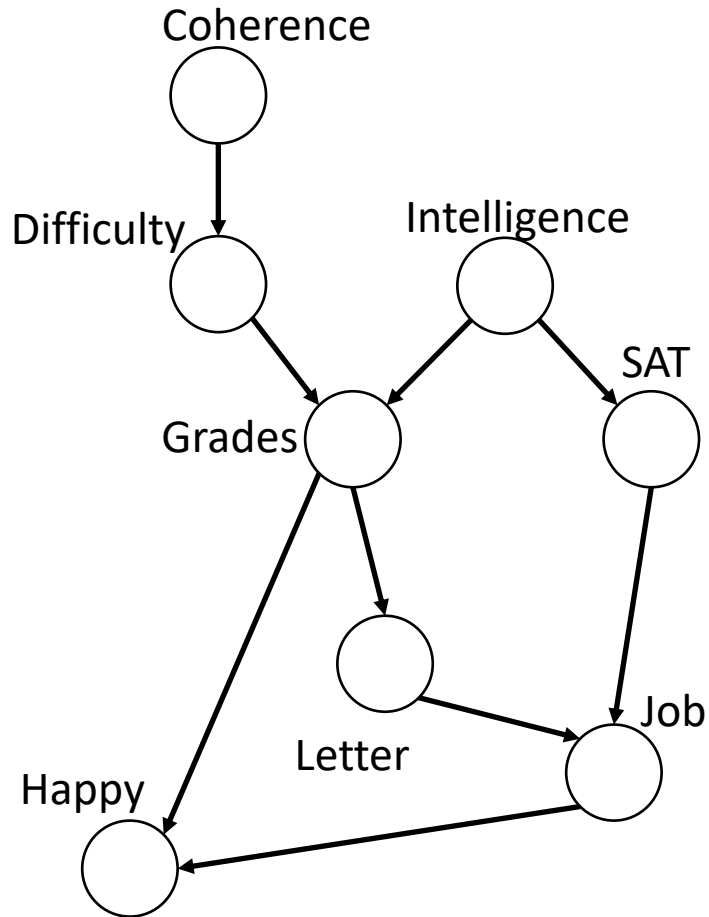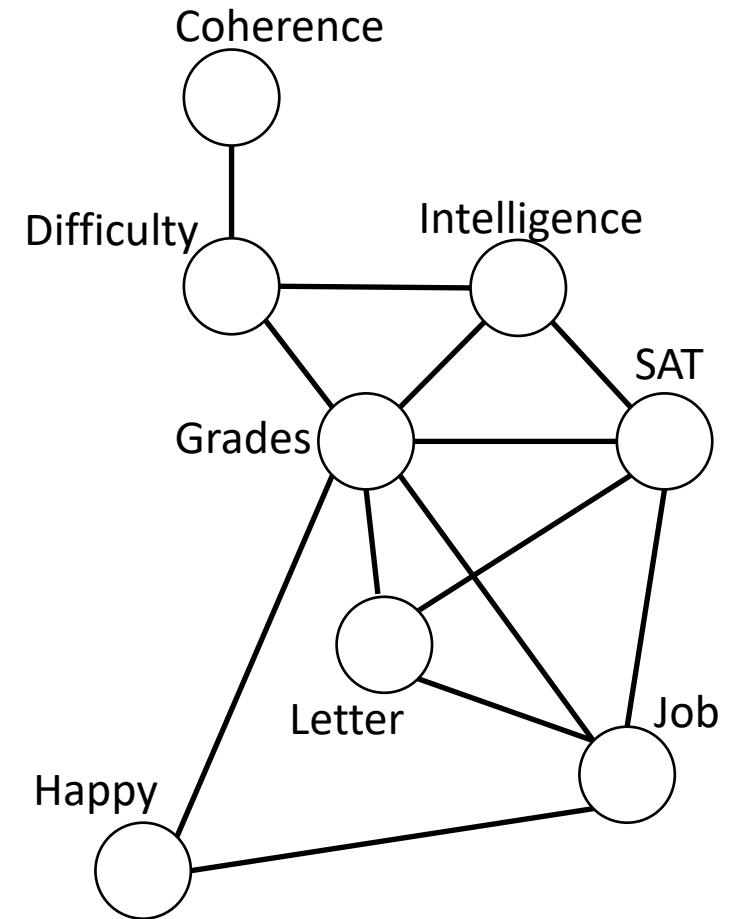Let's pick an ordering for the variable elimination
C, D, I, H, G, S, L

The rest is obvious

Coherence

Difficulty

Intelligence

SAT

Grades
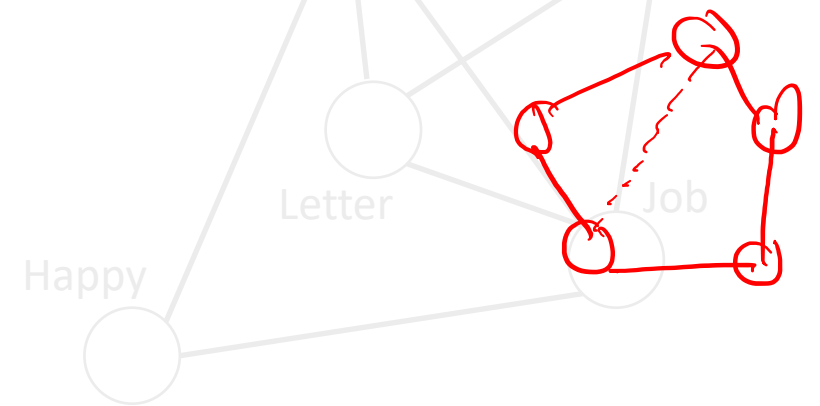
Letter

Job

Happy

OK, what we got so far?

We started with

Moralized and Triangulated

Def: An undirected graph is triangulated if every loop of length 4 or more has a *chord*. An equivalent term is that the graph is *decomposable* or *chordal*. From this definition, one may show that an undirected graph is triangulated if and only if its clique graph has a junction tree.
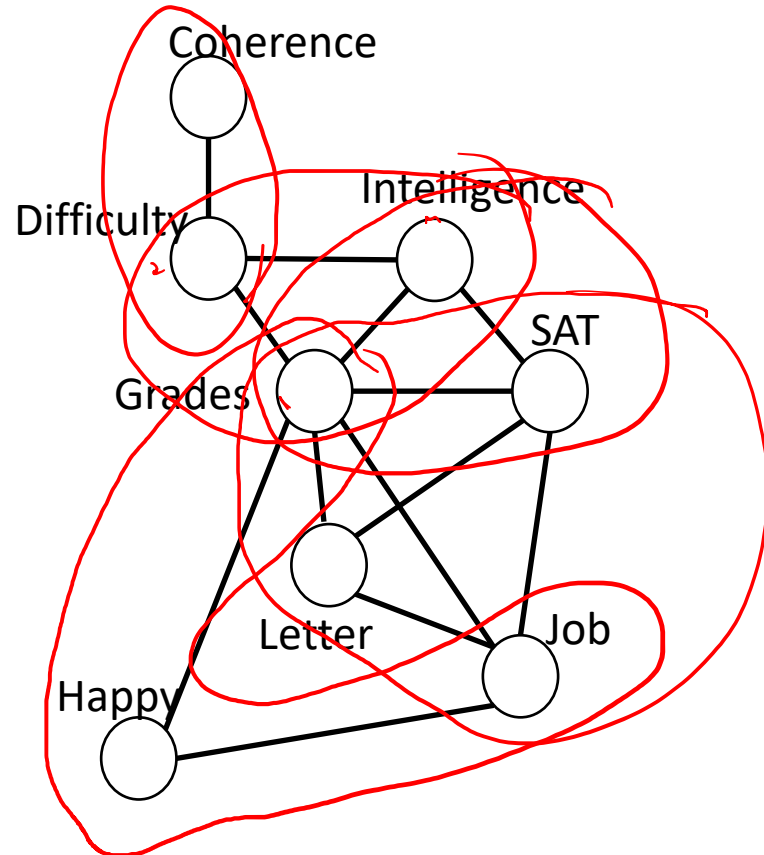
We started with
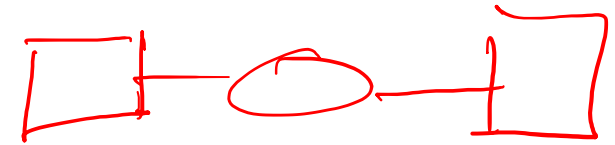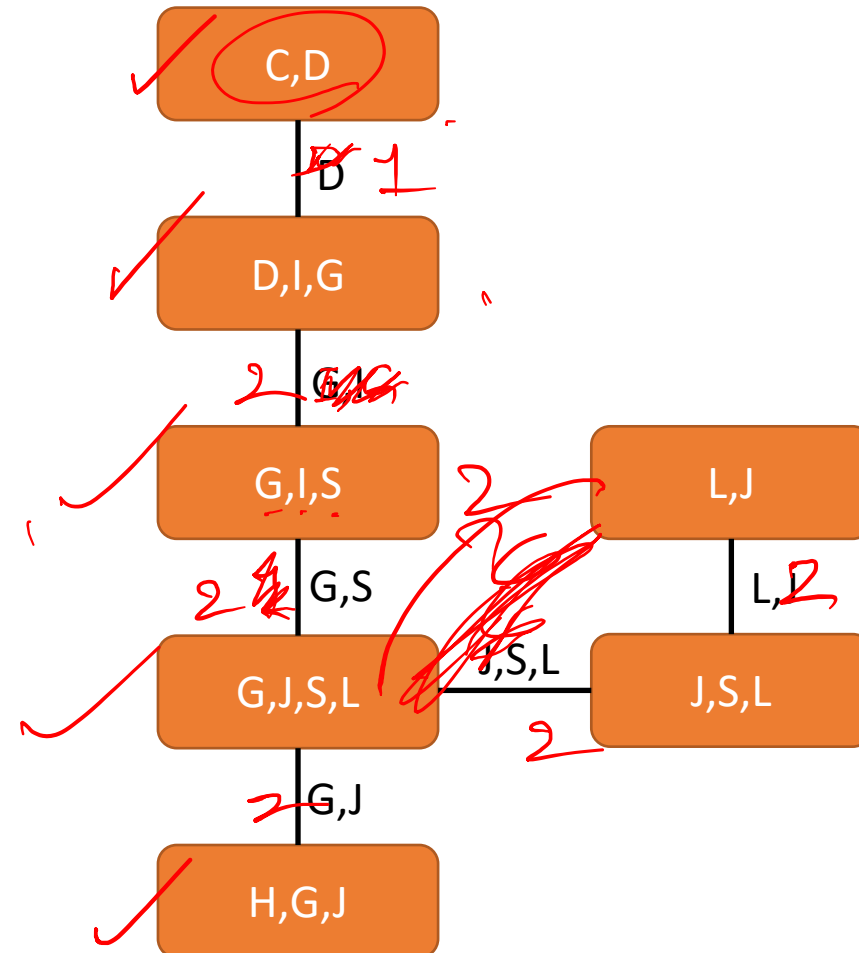
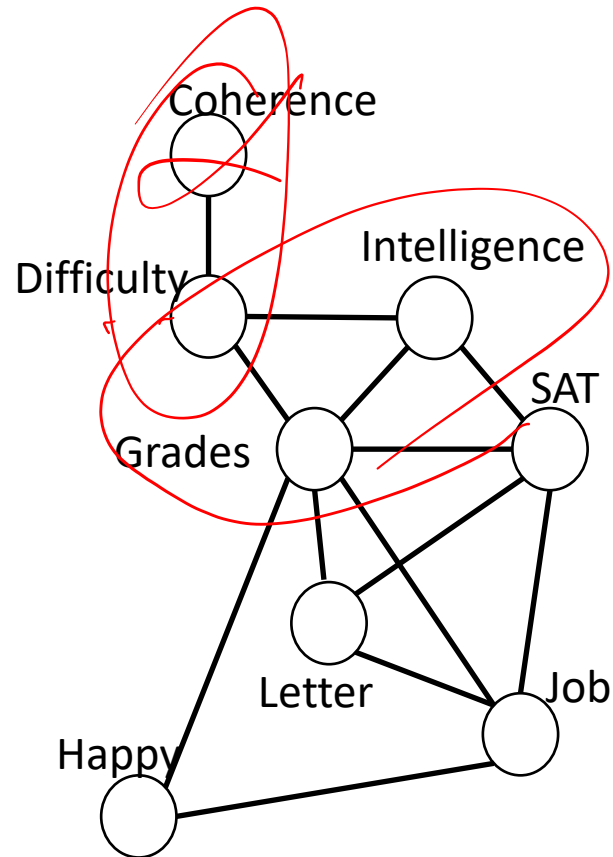Moralized and Triangulated

48

Let's build the Junction Tree

Coherence

Difficulty

Intelligence

SAT

Grades

Letter

Job

Happy

The ordering
C, D, I, H, G, S, L

C,D

D

D,I,G

G

G,I,S

G,S

G,J,S,L

J,S,L

L,J

L,J

J,S,L

G,J

H,G,J

Pass the messages on the JT

Coherence

Difficulty

Intelligence

Grades

SAT

Letter

Job

Happy

C,D

$$m_C(D) = \sum_C P(C)P(D|C)$$

D

D,I,G

$$m_D(G,I) = \sum_D m_C(D)P(G|D,I)$$

G,I

G,I,S

L,J

$$m_I(G,S) = \sum_I m_D(G,I)P(G|I)P(S|I)$$

G,S

L,J

$$m_S(J,L) = \sum_S m_G(J,S,L)P(J|L,S)$$

J,S,L

G,J,S,L

J,S,L

$$m_H(G,J) = \sum_H P(H|G,J)$$

G,J

H,G,J

The ordering
C, D, I, H, G, S, L

can you figure out the directionality?
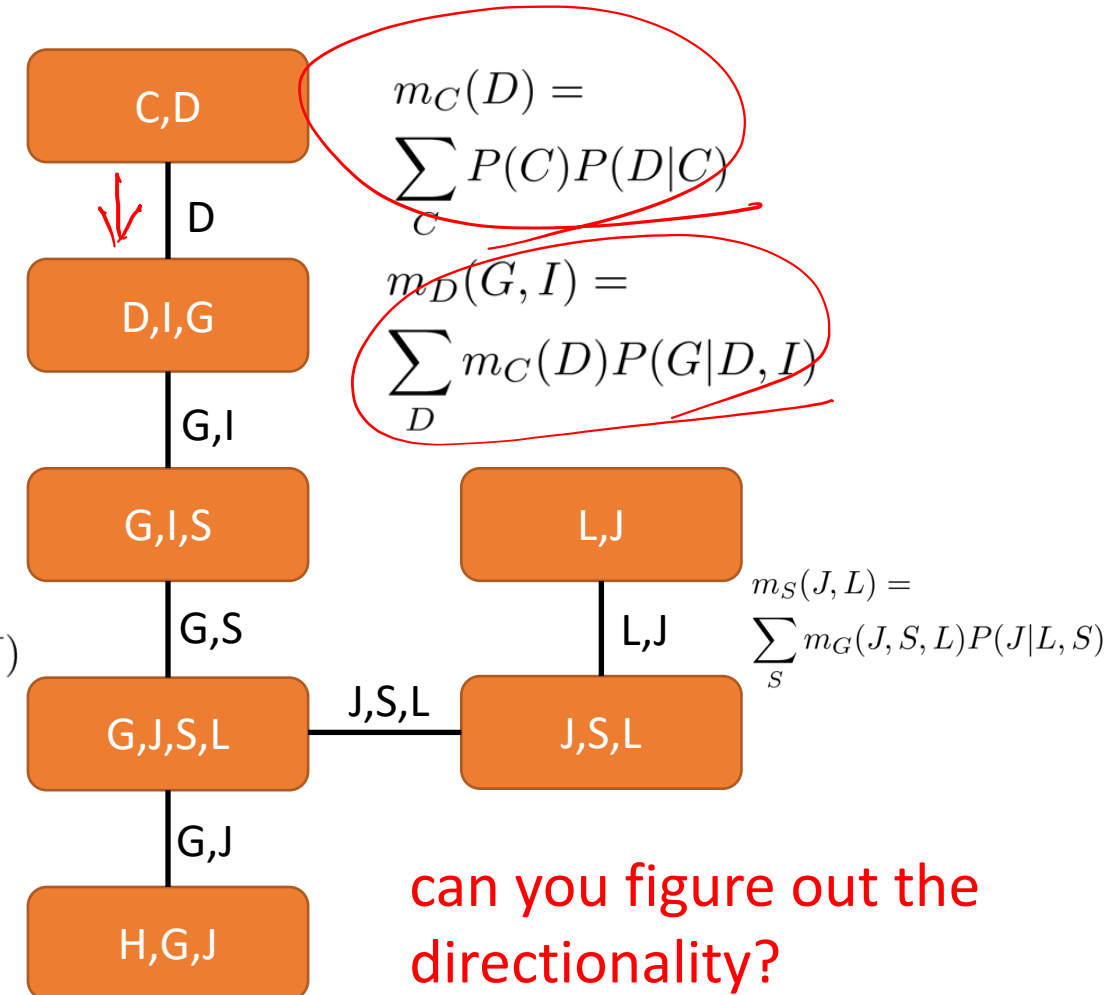
The message passing protocol

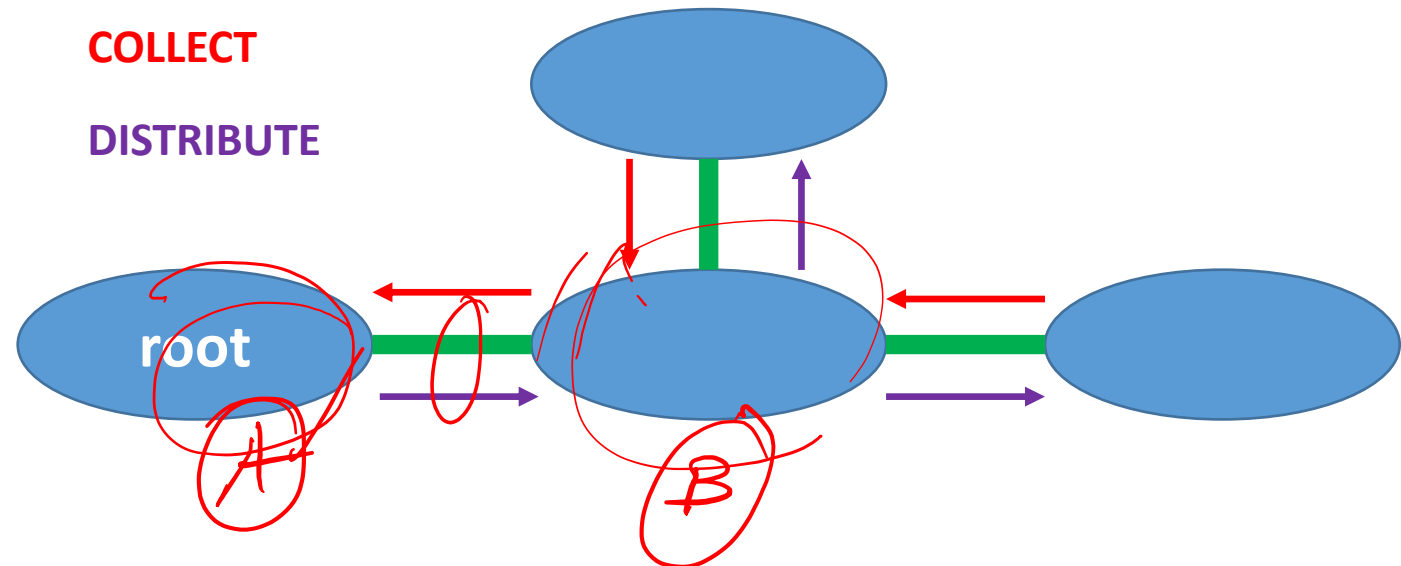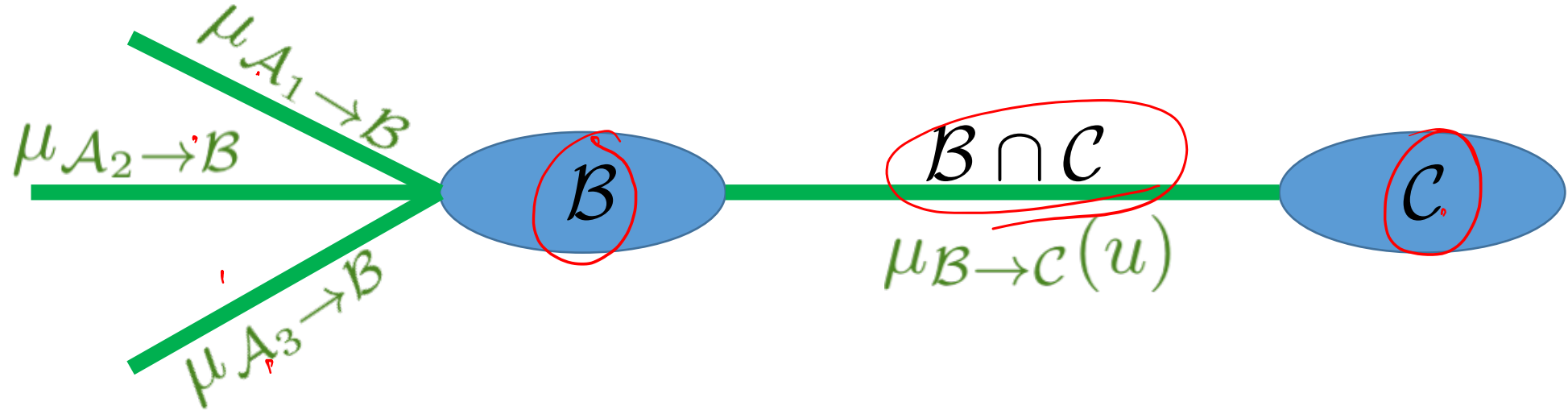> Cluster B is allowed to send a message to a neighbor C only after it has received messages from all neighbors except C.

```python
def COLLECT(C):
    for B in children (C):
        COLLECT(B)
        send message to C


def DISTRIBUTE(C):
    for B in children (C):
        send message to B
        DISTRIBUTE(C)
```

COLLECT

DISTRIBUTE

root

51

Message from Clique to another
(The Shafer-Shenoy Algorithm)

$$\mu_{\mathcal{B} \rightarrow \mathcal{C}}(u) = \sum_{v \in \mathcal{B} \backslash \mathcal{C}} \psi_{\mathcal{B}}(u \cup v) \prod_{\substack{(\mathcal{A},\mathcal{B}) \in \mathcal{E} \\ \mathcal{A} \neq \mathcal{C}}} \mu_{\mathcal{A} \rightarrow \mathcal{B}}(u_A \cup v_A)$$

Formal Algorithm

- **Moralisation**: Marry the parents (only for directed distributions).

- **Triangulation**: Ensure that every loop of length 4 or more has a chord.

- **Junction Tree**: Form a junction tree from cliques of the triangulated graph, removing any unnecessary links in a loop on the cluster graph. Algorithmically, this can be achieved by finding a tree with *maximal spanning weight* with weight given by the number of variables in the separator between cliques. Alternatively, given a clique *elimination order* (with the lowest cliques eliminated first), one may connect each clique to the single neighboring clique.

- **Potential Assignment**: Assign potentials to junction tree cliques and set the separator potentials to unity.

- **Message Propagation**

Some Facts about BP

$$f(x) = x$$

- BP is exact on trees.

- If BP converges it has reached a local minimum of an objective function
- (the Bethe free energy *Yedidia et.al '00 , Heskes '02*)→*often good approximation*

- *If* it converges, convergence is fast near the fixed point.
- Many exciting applications:
  - error correcting decoding (*MacKay, Yedidia, McEliece, Frey*)
  - vision (*Freeman, Weiss*)
  - bioinformatics (*Weiss*)
  - constraint satisfaction problems (*Dechter*)
  - game theory (*Kearns*)
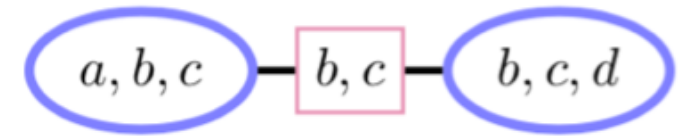 - …

Summary of the Network Zoo
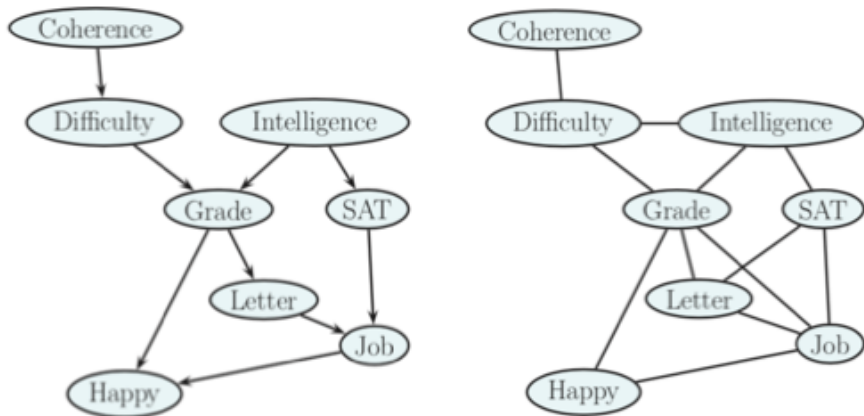
UGM and DGM
- Use to represent family of probability distributions
- Clear definition of arrows and circles
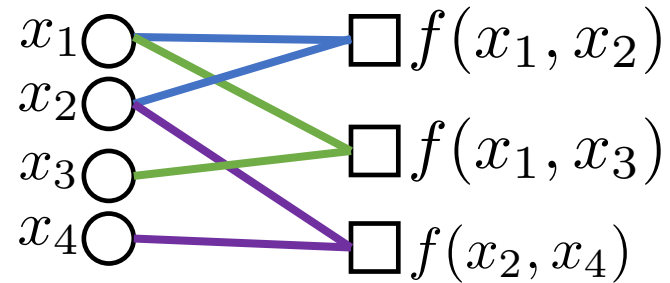
Factor Graph
- A way to present factorization for both UGM and DGM
- It is bipartite graph
- More like a data structure
- Not to read the independencies

Clique graph or Junction Tree
- A data structure used for exact inference and message passing
- Nodes are cluster of variables
- Not to read the independencies