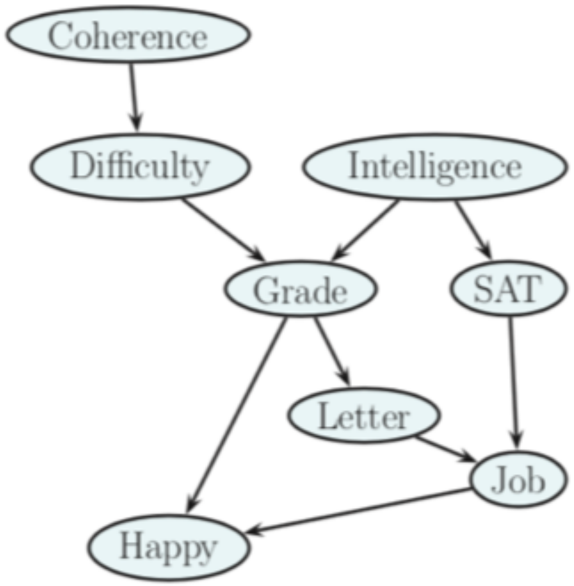
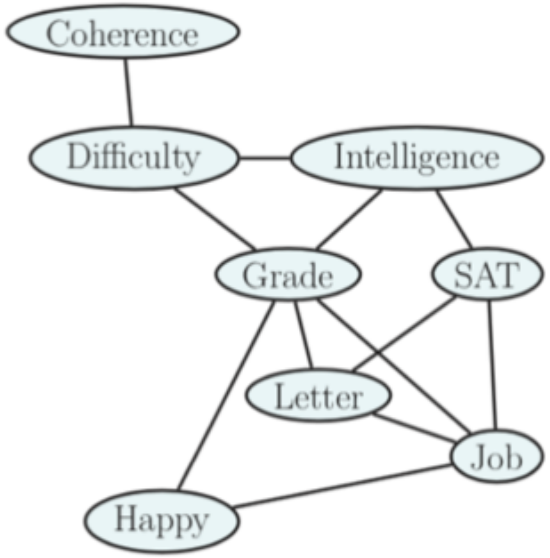
Message Passing and Junction Tree Algorithms

Kayhan Batmanghelich

Review





$$
\begin{aligned}
&P(C, D, I, G, S, L, J, H) \\
&\quad = \quad P(C)P(D|C)P(I)P(G|I, D)P(S|I)P(L|G)P(J|L, S)P(H|G, J)
\end{aligned}
$$

$$
\begin{aligned}
&p(C, D, I, G, S, L, J, H) \\
&\quad = \quad \psi_C(C)\psi_D(D, C)\psi_I(I)\psi_G(G, I, D)\psi_S(S, I)\psi_L(L, G)\psi_J(J, L, S)\psi_H(H, G, J)
\end{aligned}
$$

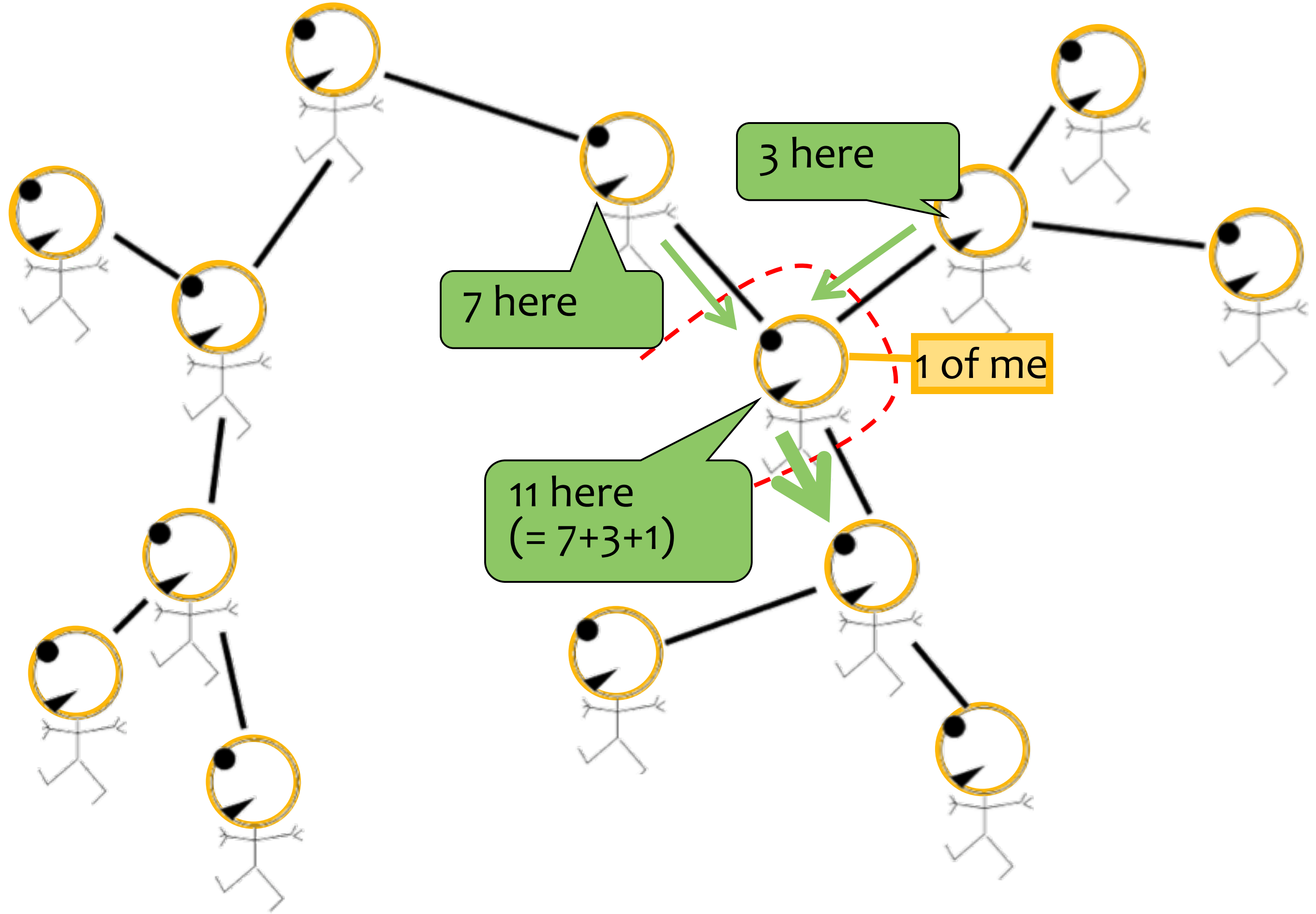
Review

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

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

Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

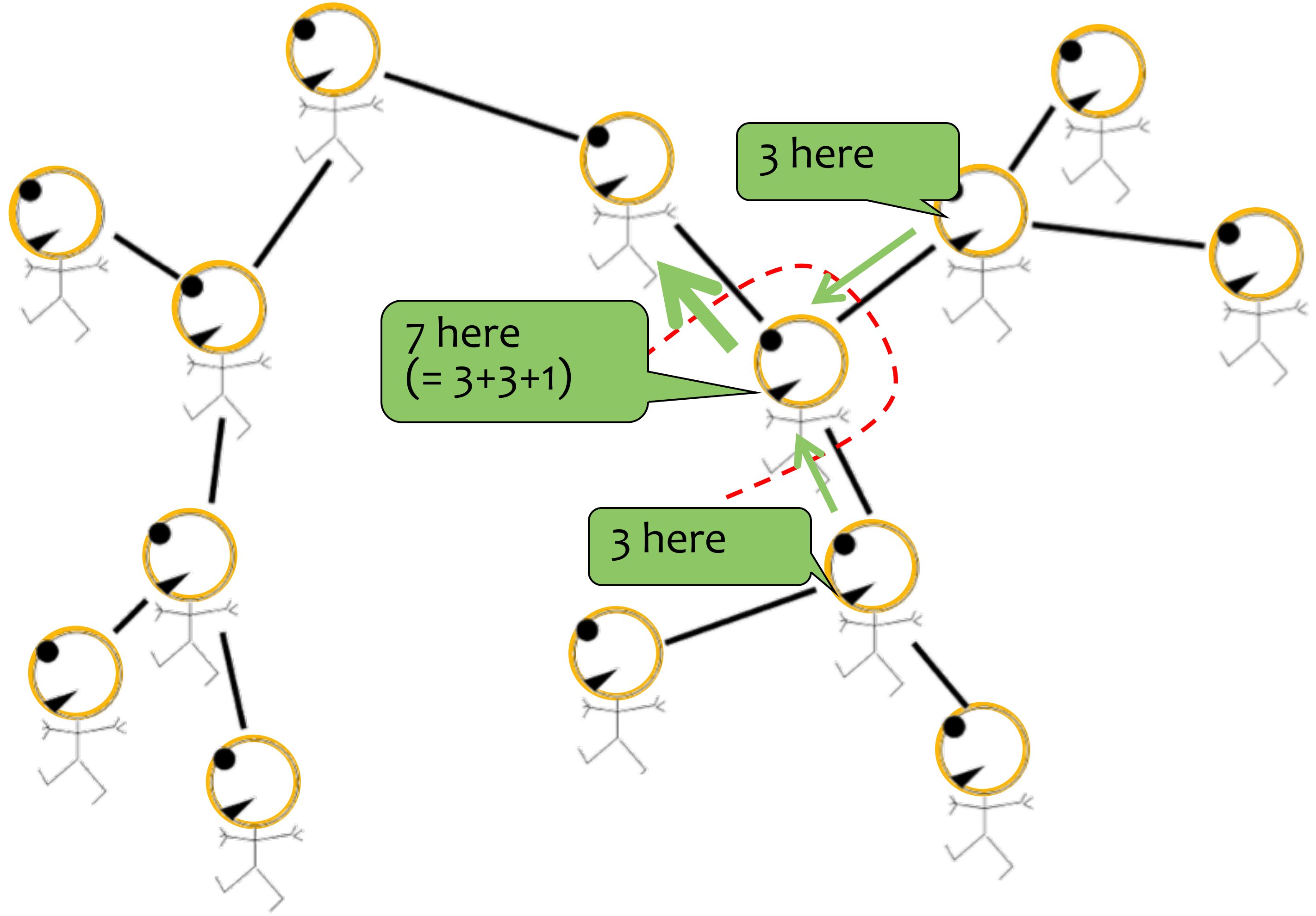


Slides adapted from
Matt Gormley (2016)

adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

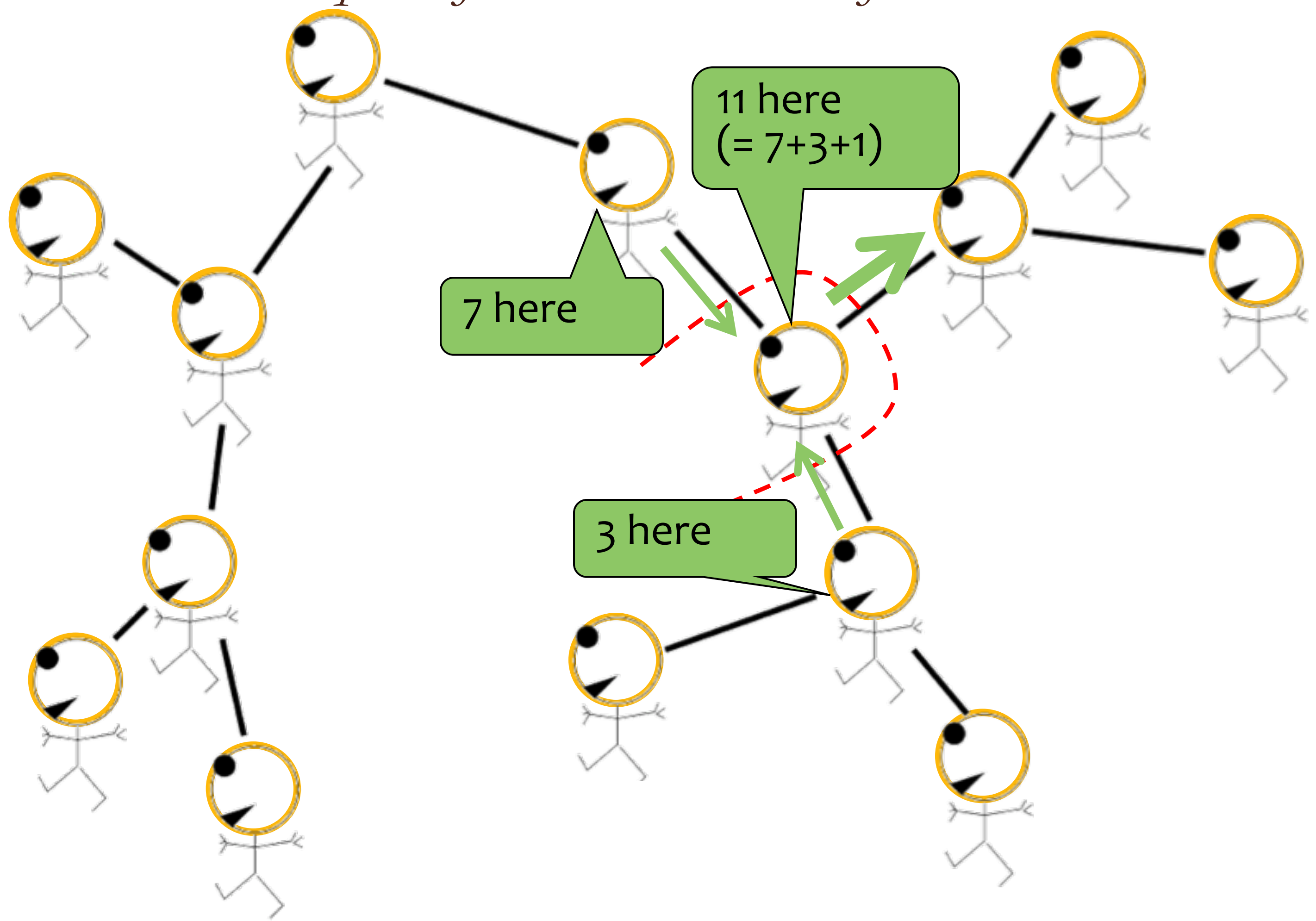


Slides adapted from
Matt Gormley (2016)

adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

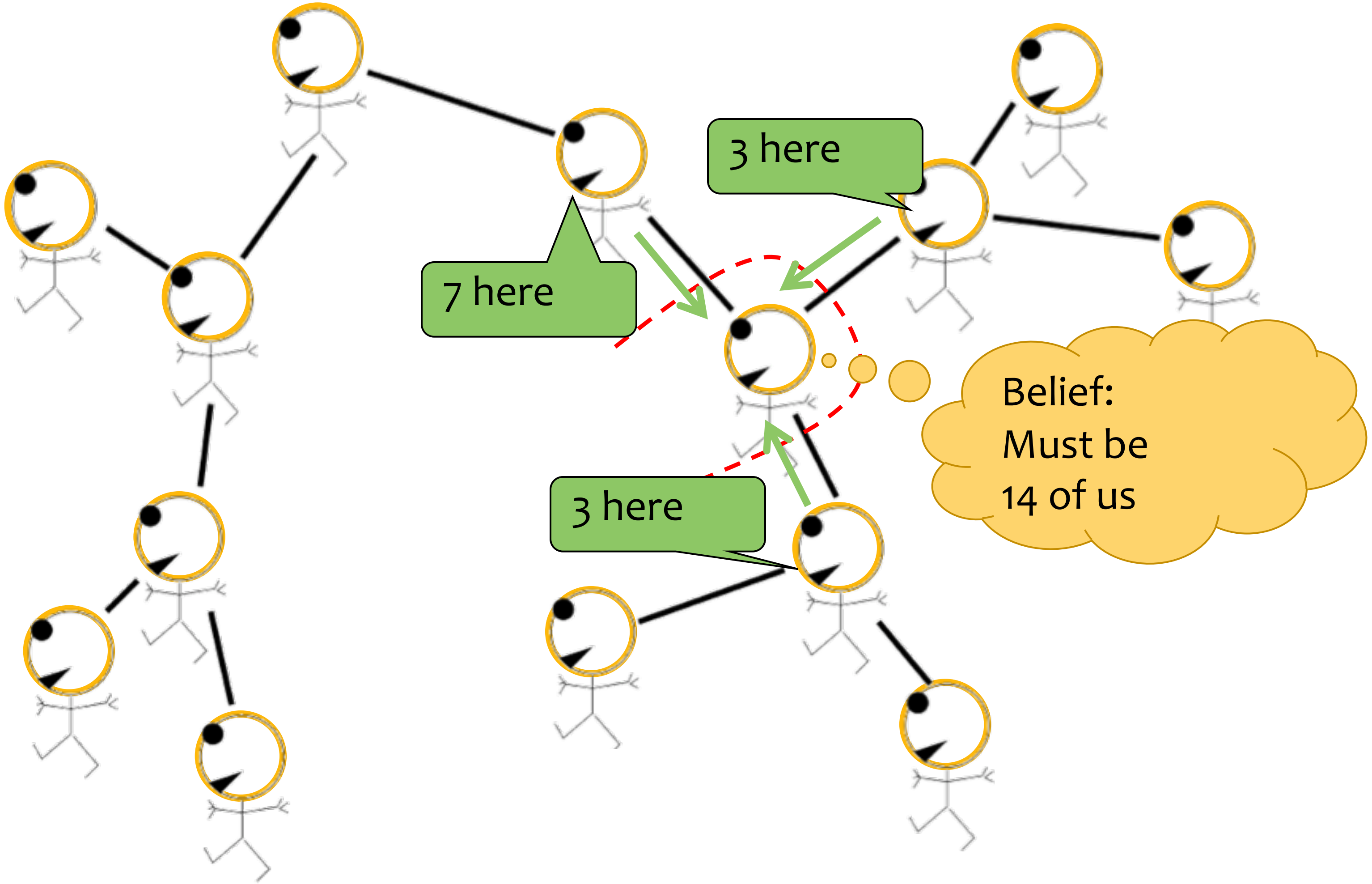


Slides adapted from
Matt Gormley (2016)

adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

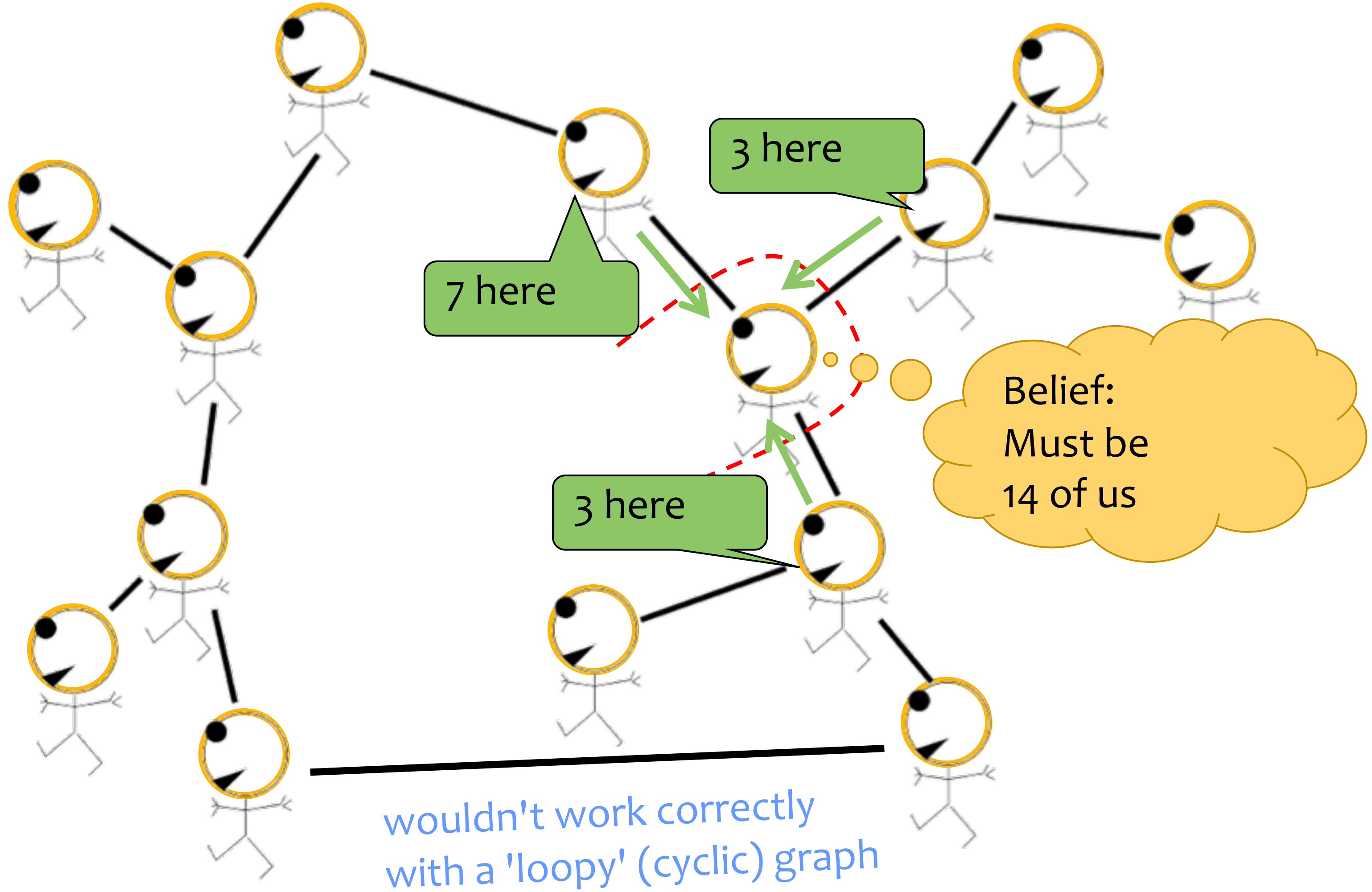


Slides adapted from
Matt Gormley (2016)

adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

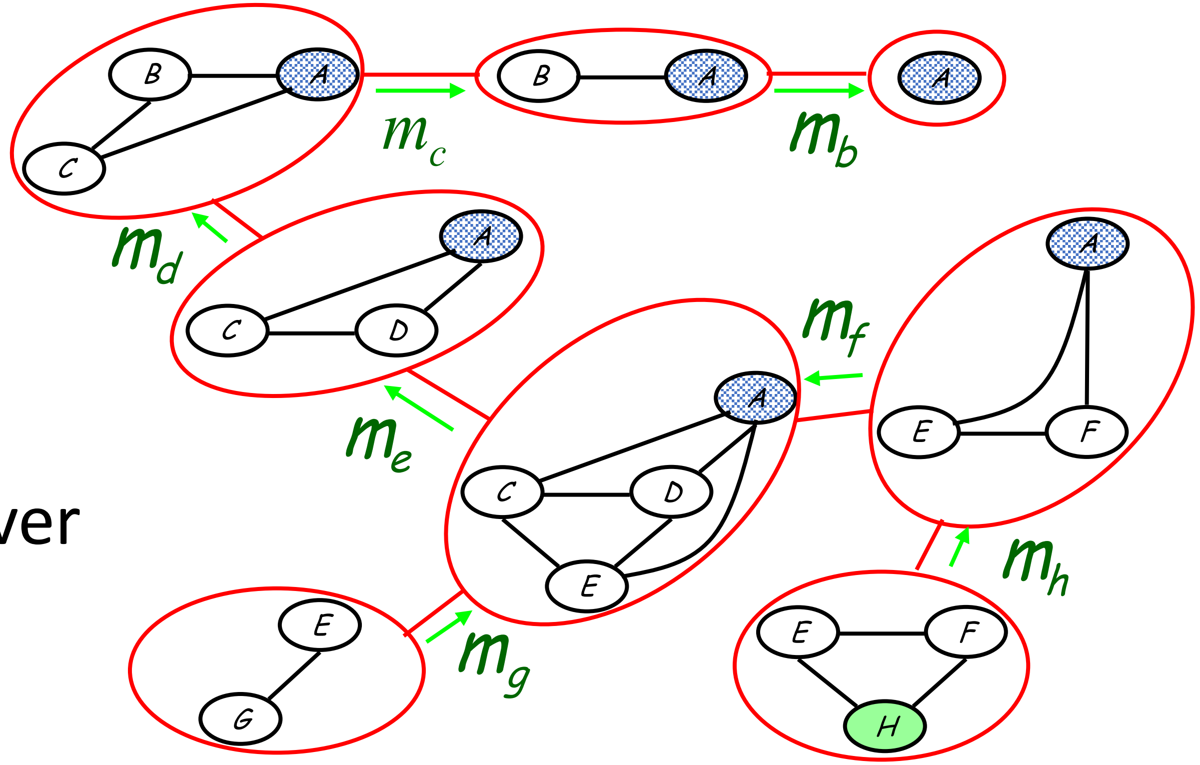
Each soldier receives reports from all branches of tree



Slides adapted from
Matt Gormley (2016)

adapted from MacKay (2003) textbook

Review



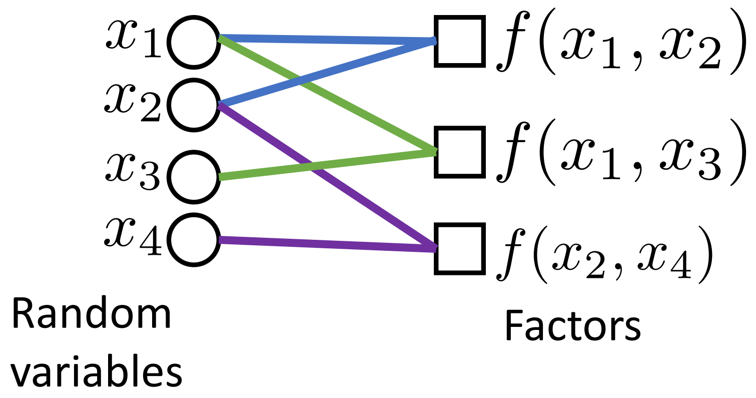
Message from one C1 to C2:

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?



- 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.

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:

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

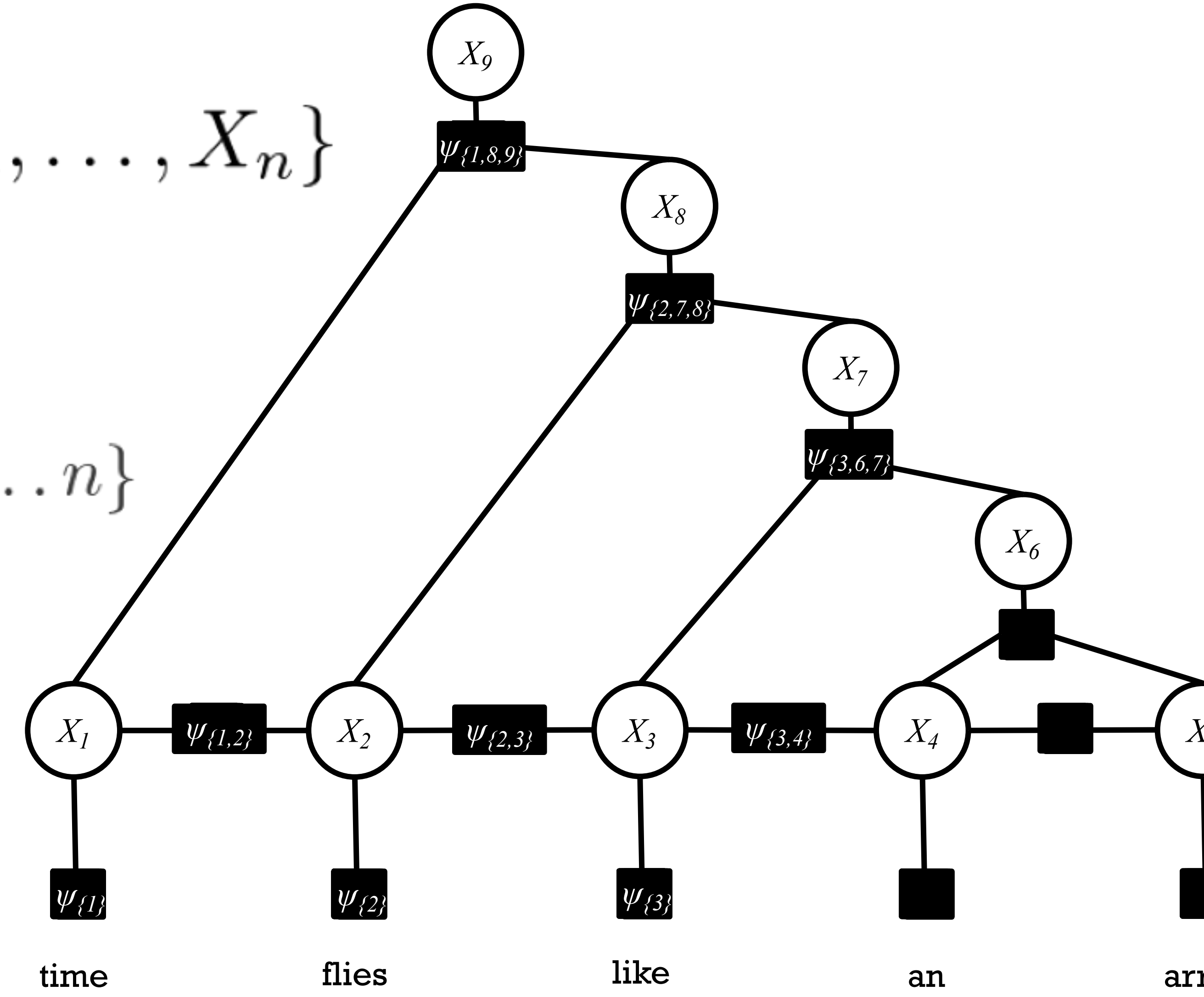
- Factors:

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

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

Joint Distribution

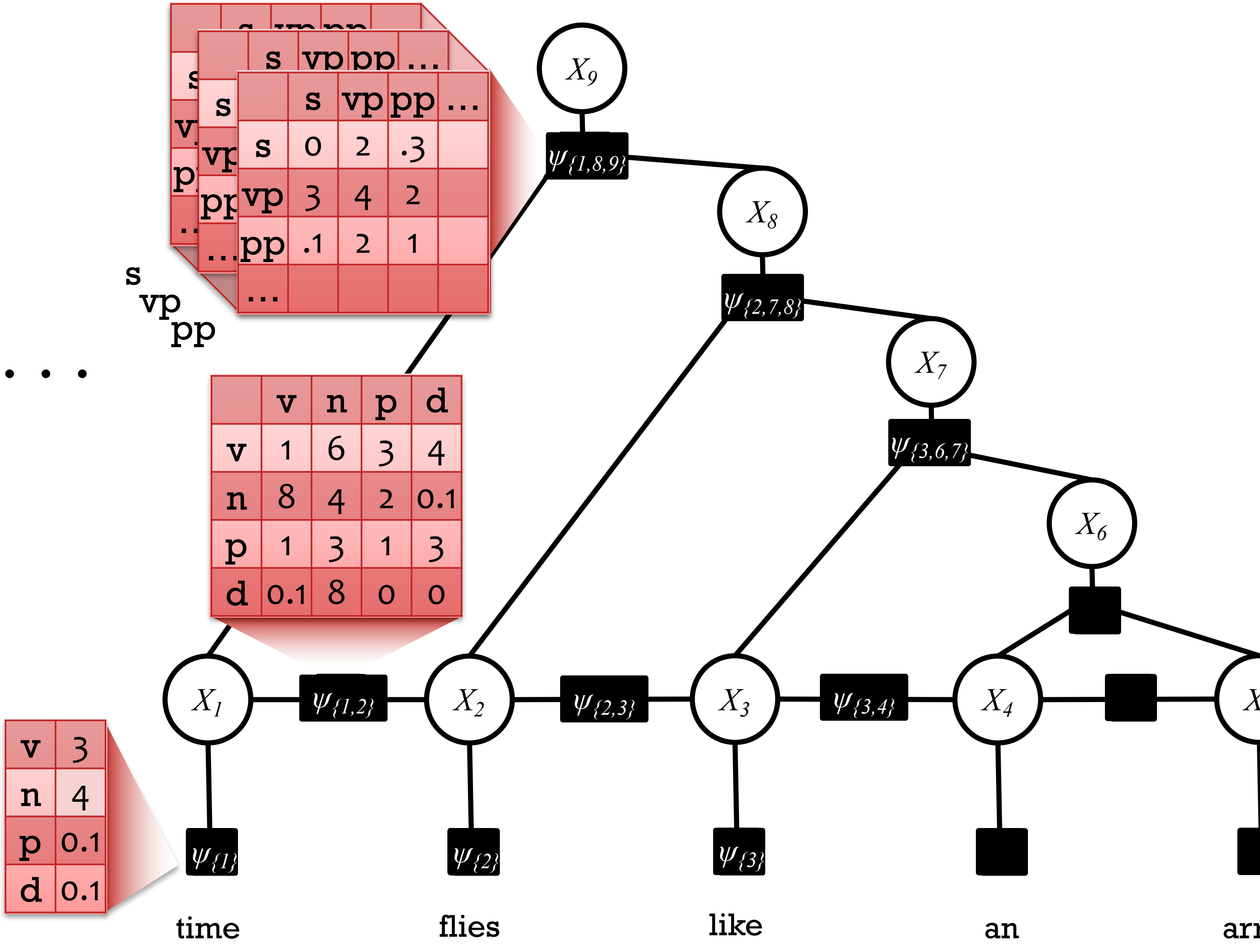
$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_\alpha \psi_\alpha(\boldsymbol{x}_\alpha)$$



Factors are Tensors

- **Factors:**

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



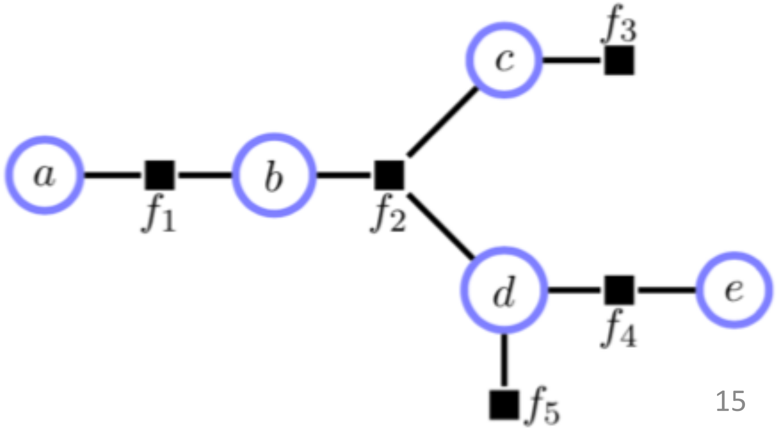
An Inference Example

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

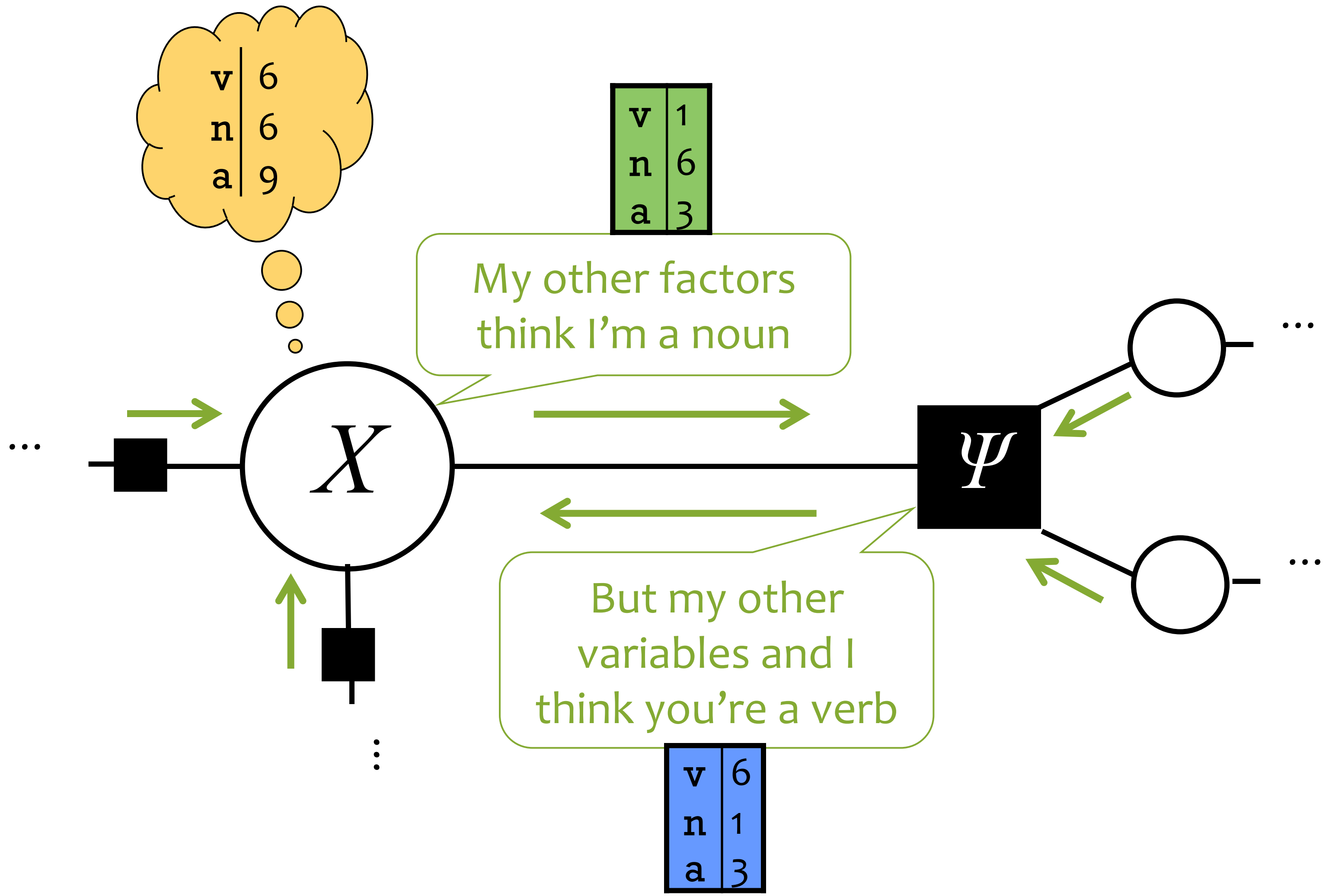
$$p(a|b)p(b|c,d)p(c)p(d)p(e|d)$$

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

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



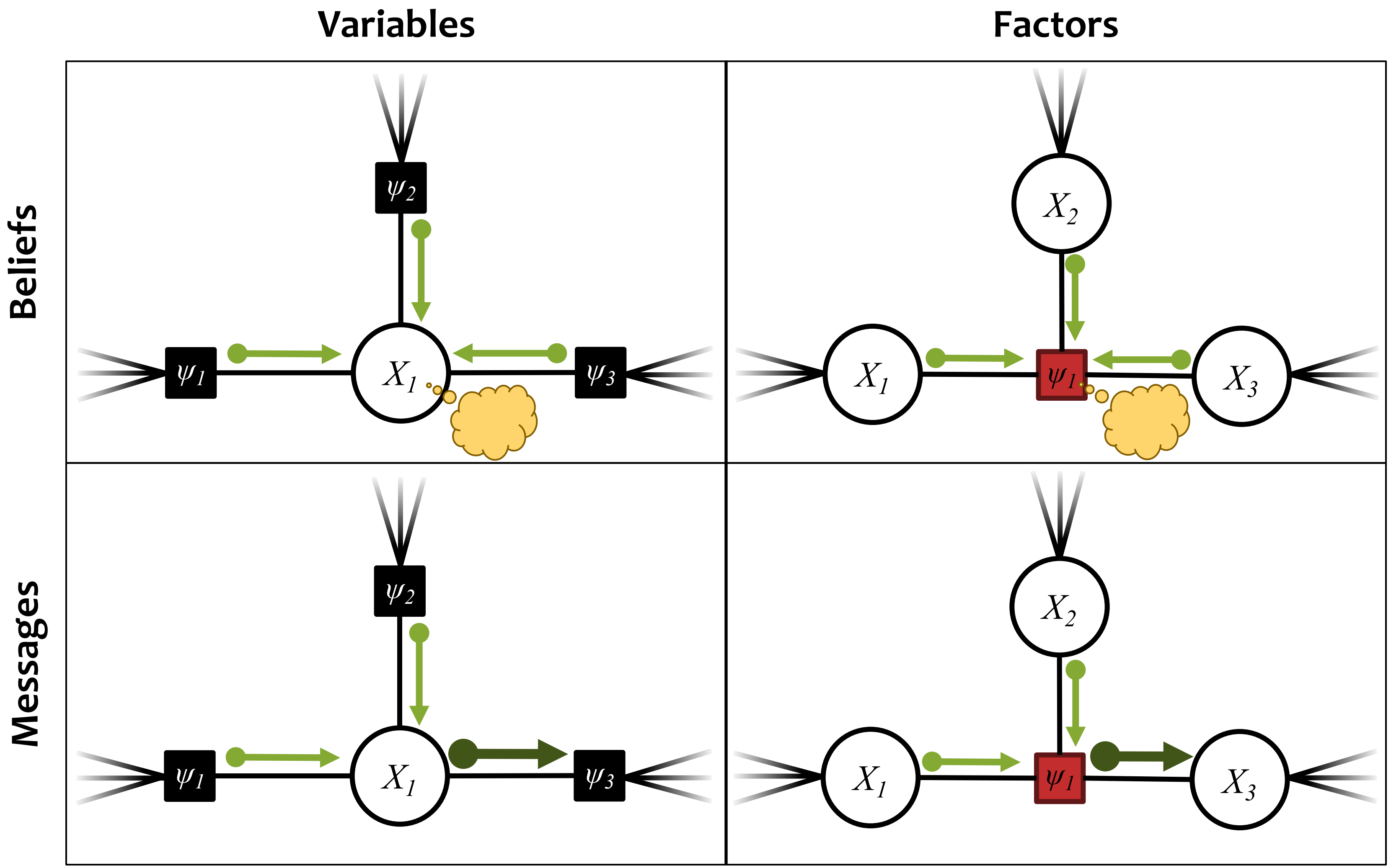
Message Passing in Belief Propagation



Slides adapted from Matt Gormley (2016)

Both of these messages judge the possible values of variable X.
Their product = belief at X = product of all 3 messages to X.

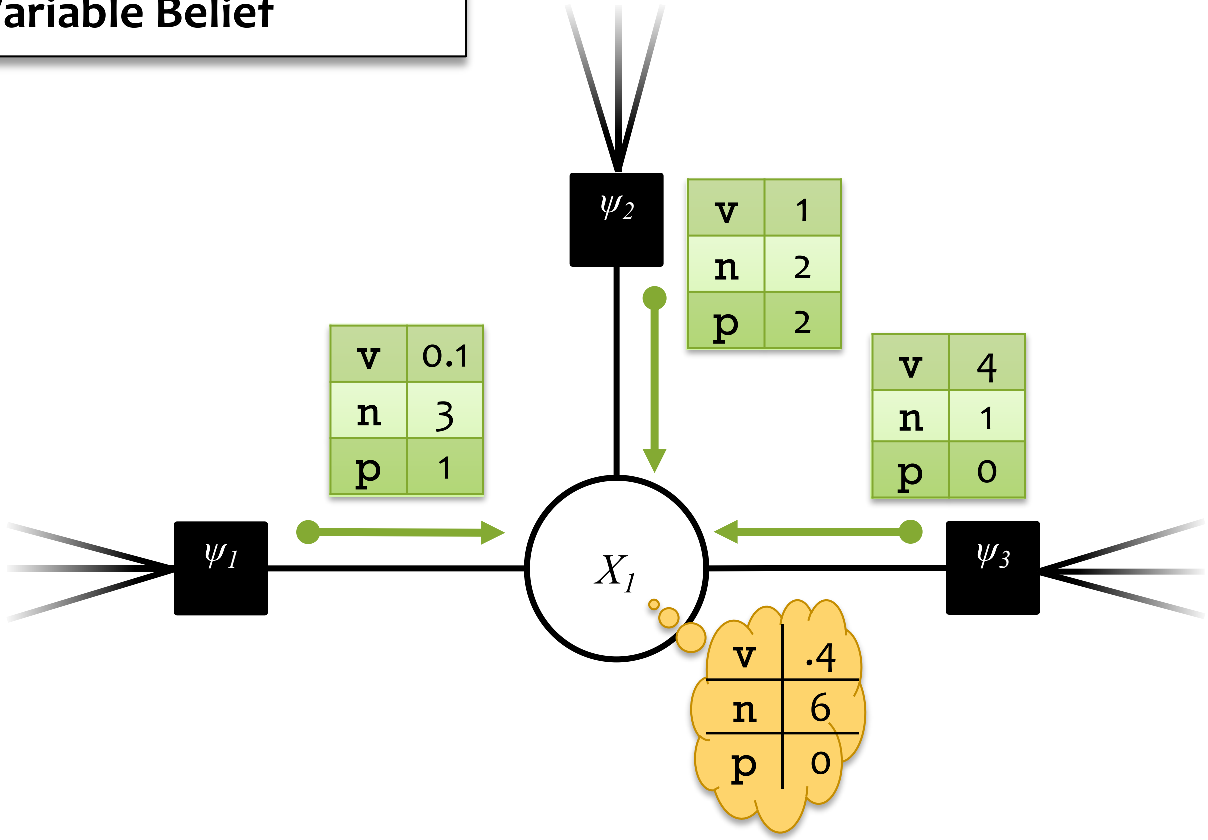
Sum-Product Belief Propagation



Slides adapted from
Matt Gormley (2016)

Sum-Product Belief Propagation

Variable Belief

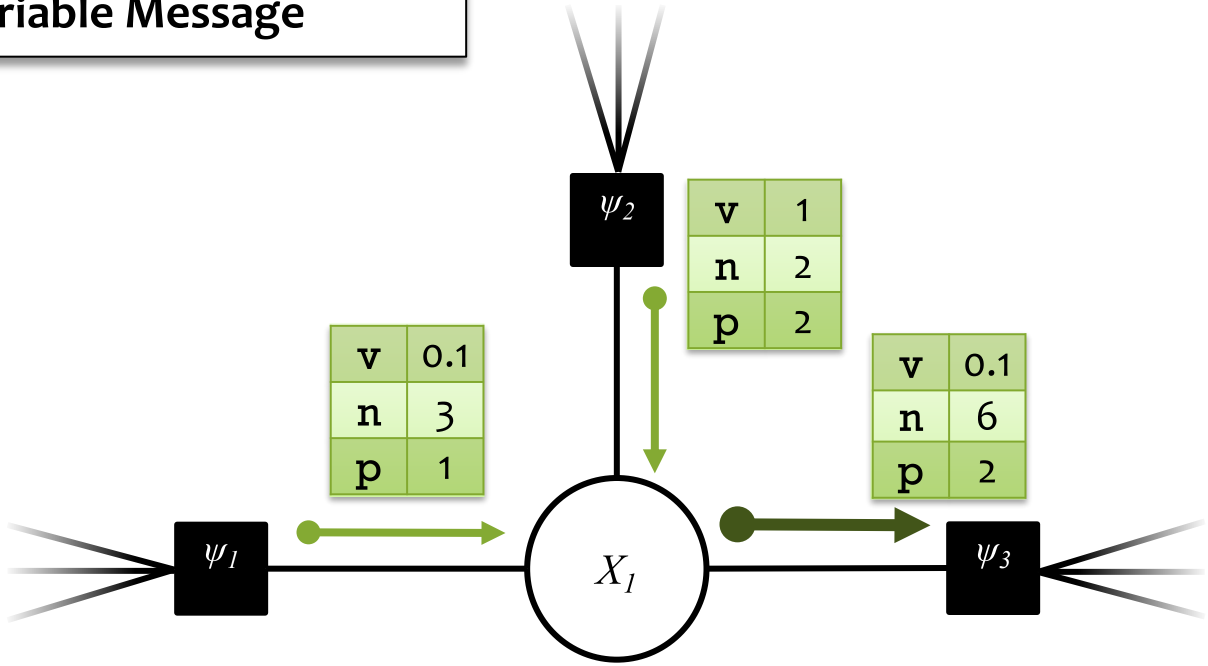


Slides adapted from Matt Gormley (2016)

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

Sum-Product Belief Propagation

Variable Message

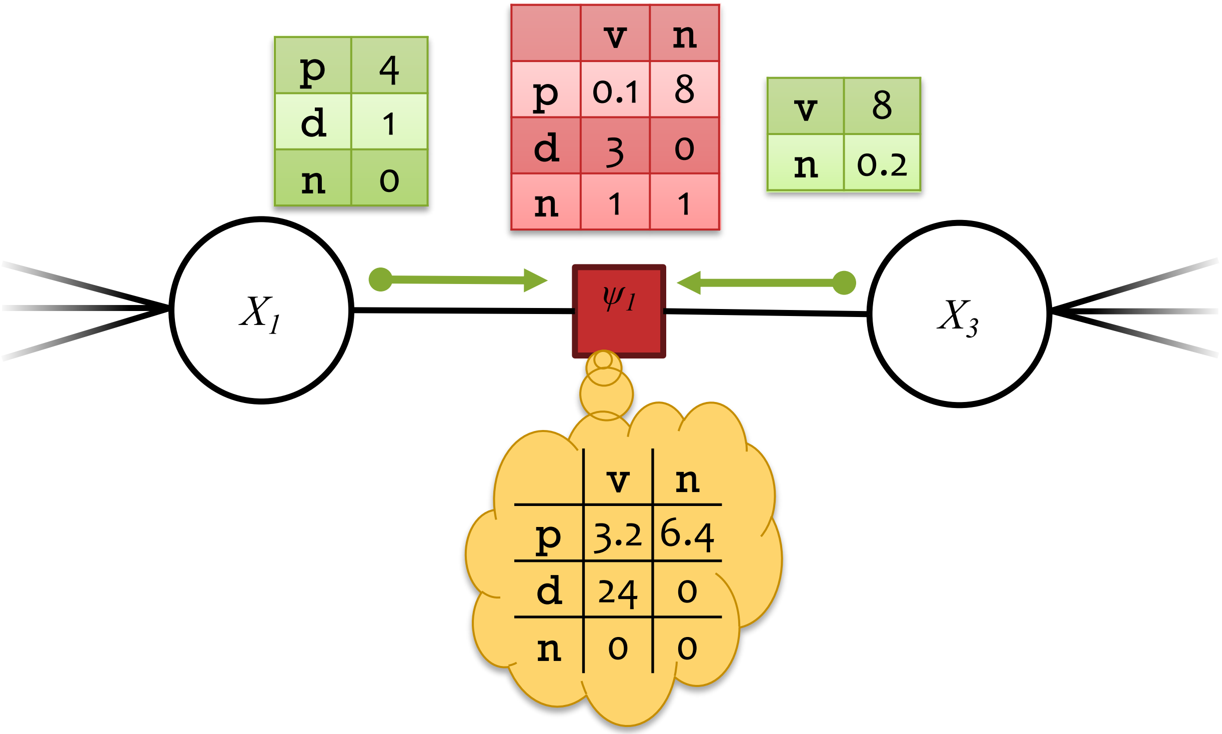


Slides adapted from
Matt Gormley (2016)

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

Sum-Product Belief Propagation

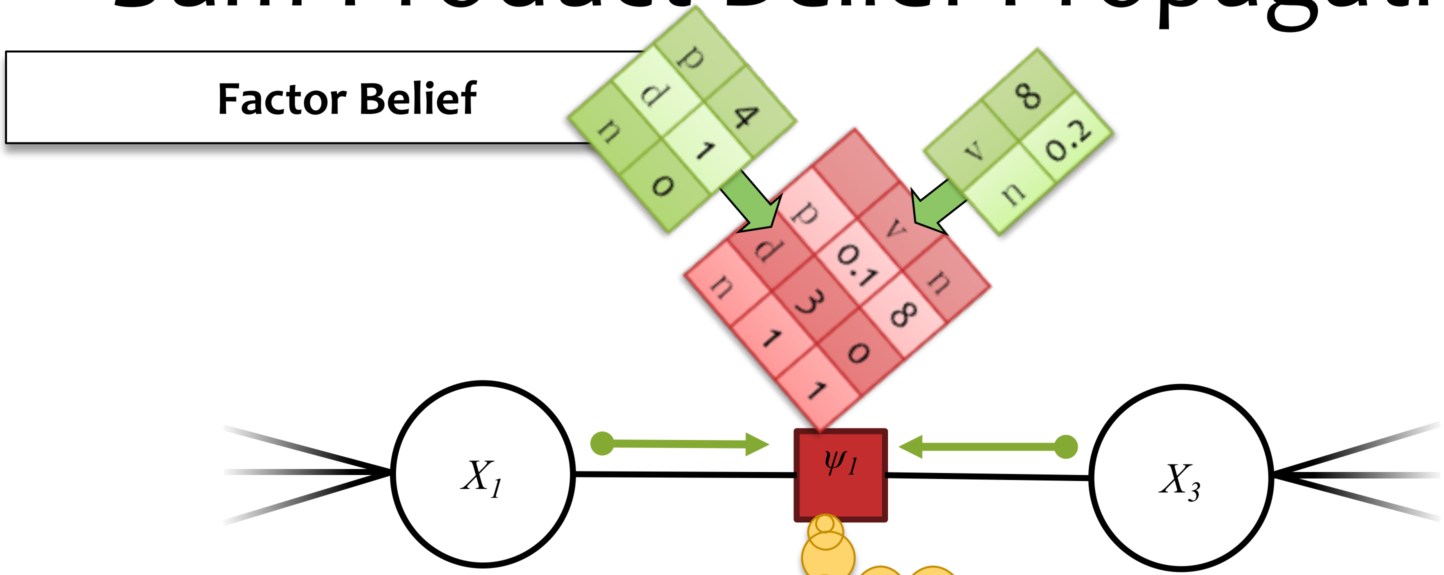
Factor Belief



Slides adapted from Matt Gormley (2016)

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

Sum-Product Belief Propagation



	v	n
p	3.2	6.4
d	24	0
n	0	0

Slides adapted from Matt Gormley (2016)

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

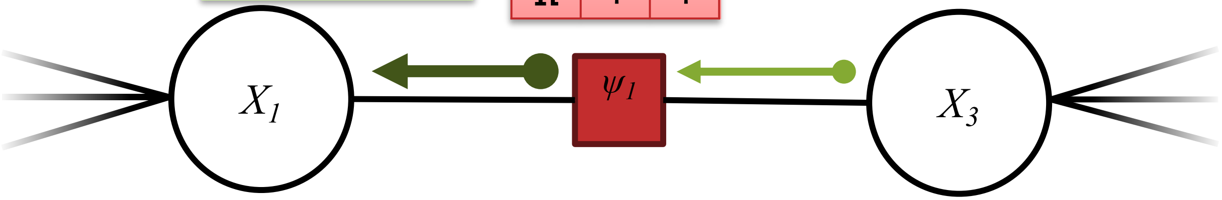
Sum-Product Belief Propagation

Factor Message

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

	v	n
p	0.1	8
d	3	0
n	1	1

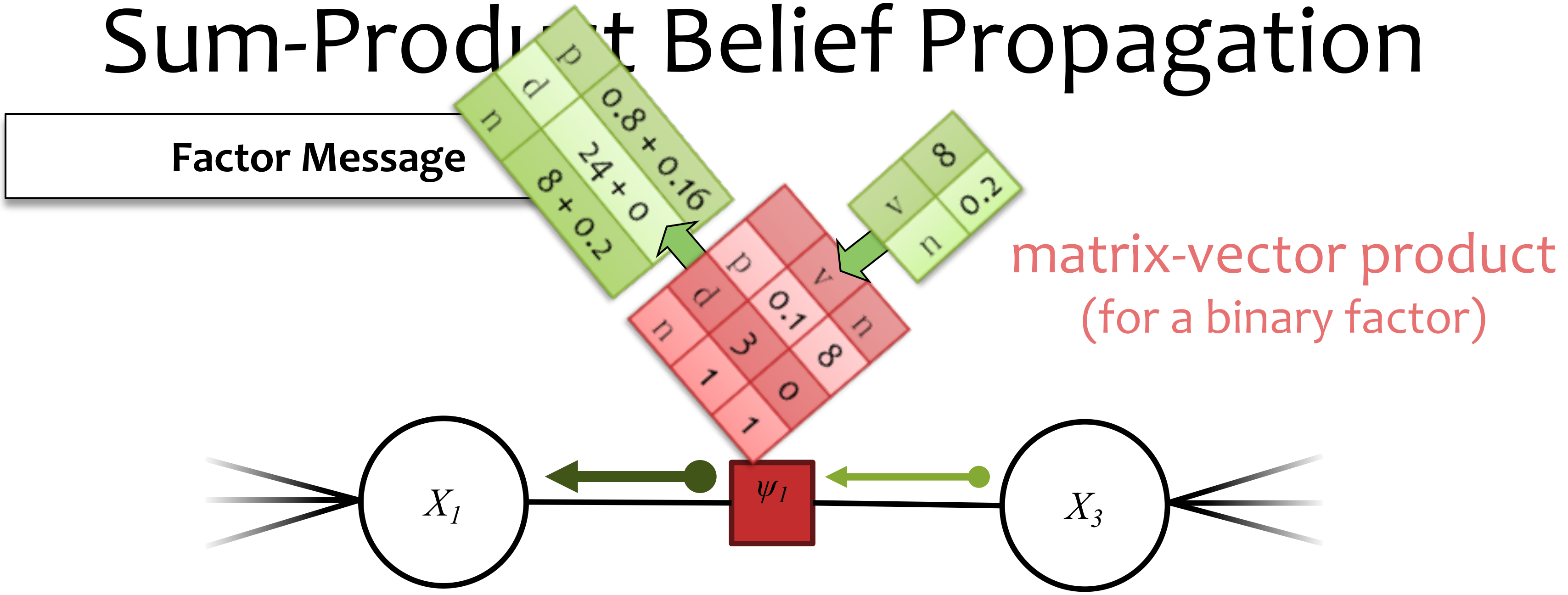
v	8
n	0.2



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

Slides adapted from
Matt Gormley (2016)

Sum-Product Belief Propagation

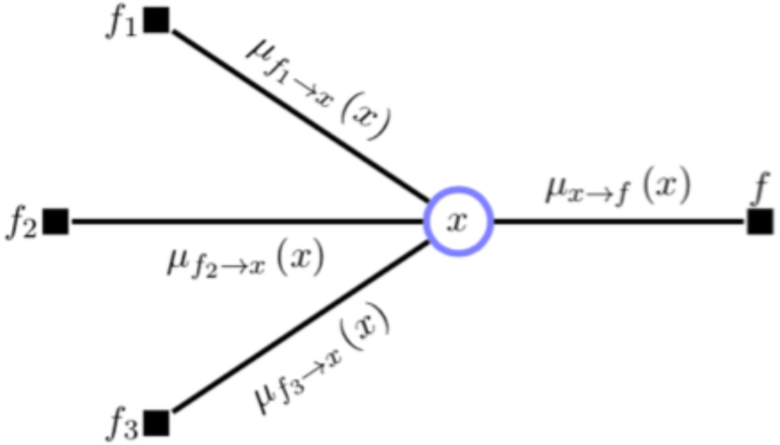


$$\mu_{\alpha \to i}(x_i) = \sum_{\boldsymbol{x_\alpha} : \boldsymbol{x_\alpha}[i] = x_i} \psi_\alpha(\boldsymbol{x_\alpha}) \prod_{j \in \mathcal{N}(\alpha) \setminus i} \mu_{j \to \alpha}(\boldsymbol{x_\alpha}[i])$$

Summary of the Messages

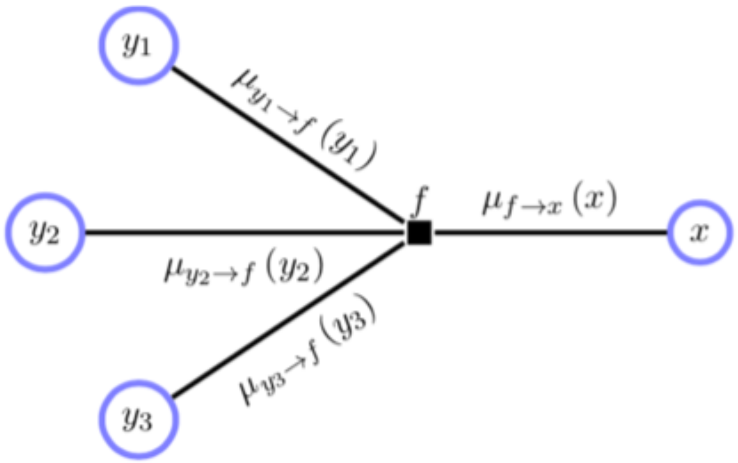
Variable to Factor message

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



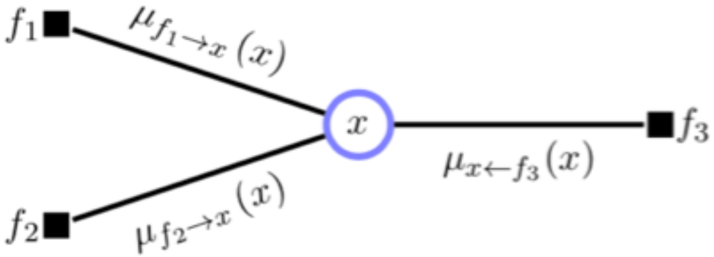
Factor to Variable message

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



Marginal

$$p(x) \propto \prod_{f\in\mathrm{ne}(x)} \mu_{f\to x}(x)$$



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.

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

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

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

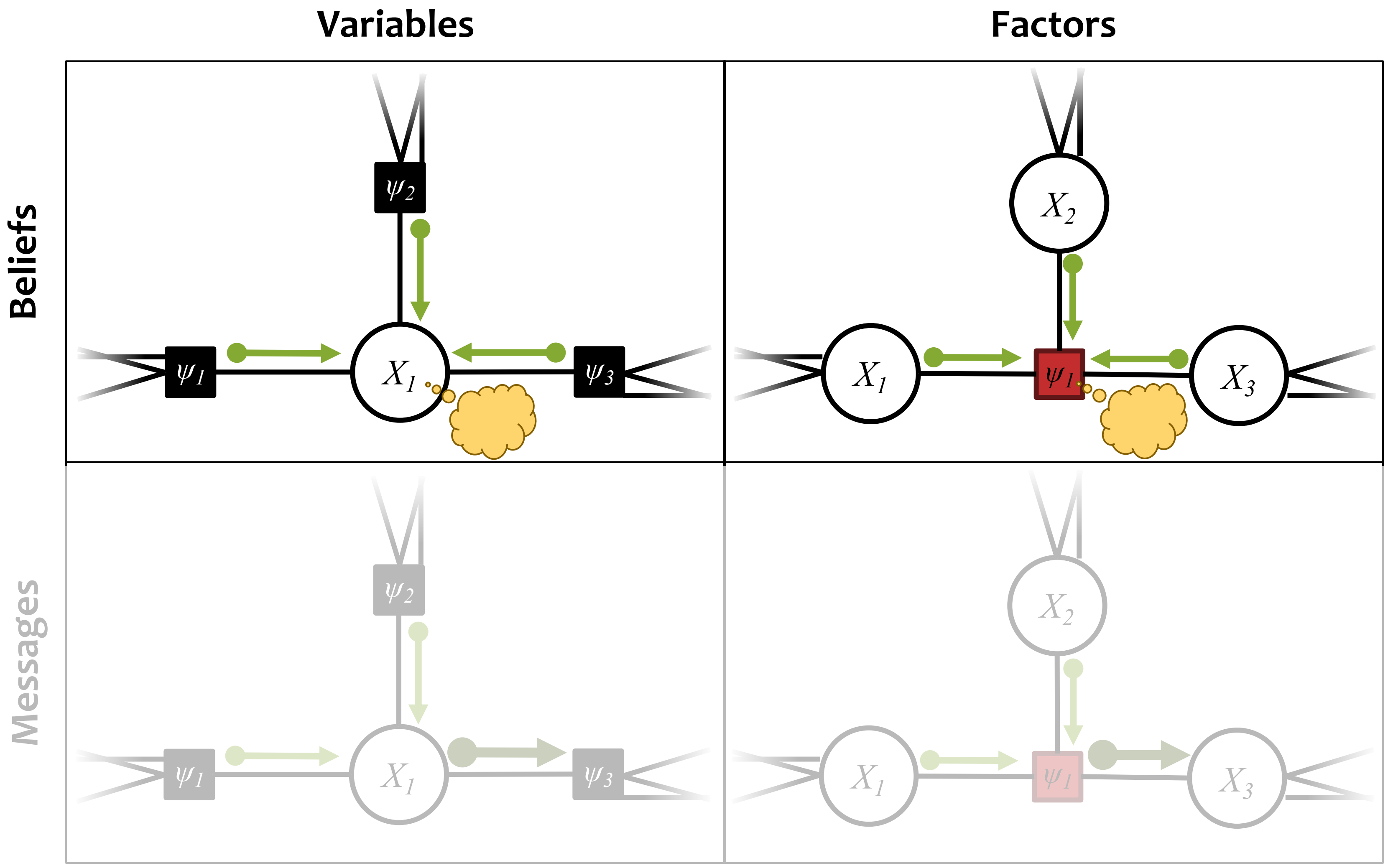
1. Compute the beliefs (unnormalized marginals).

$$\boxed{b_i(x_i) = \prod_{\alpha \in \mathcal{N}(i)} \mu_{\alpha \to i}(x_i)} \quad \boxed{b_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) = \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) \prod_{i \in \mathcal{N}(\alpha)} \mu_{i \to \alpha}(\boldsymbol{x}_{\boldsymbol{\alpha}}[i])}$$

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

$$\boxed{p_i(x_i) \propto b_i(x_i)} \quad \boxed{p_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) \propto b_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}})}$$

Sum-Product Belief Propagation

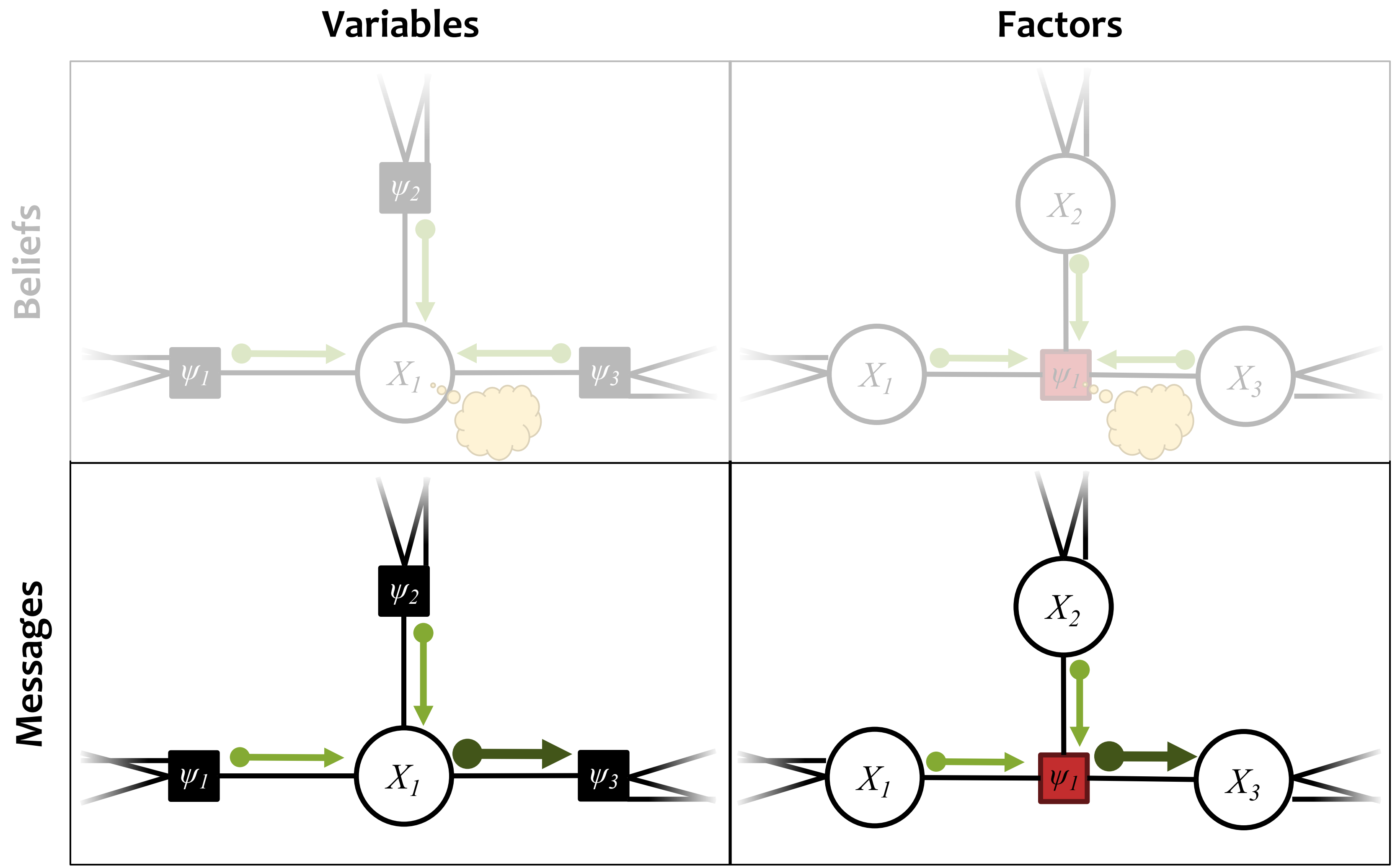


$$b_i(x_i) = \prod_{\alpha \in \mathcal{N}(i)} \mu_{\alpha \to i}(x_i)$$

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

Slides adapted from Matt Gormley (2016)

Sum-Product Belief Propagation



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

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

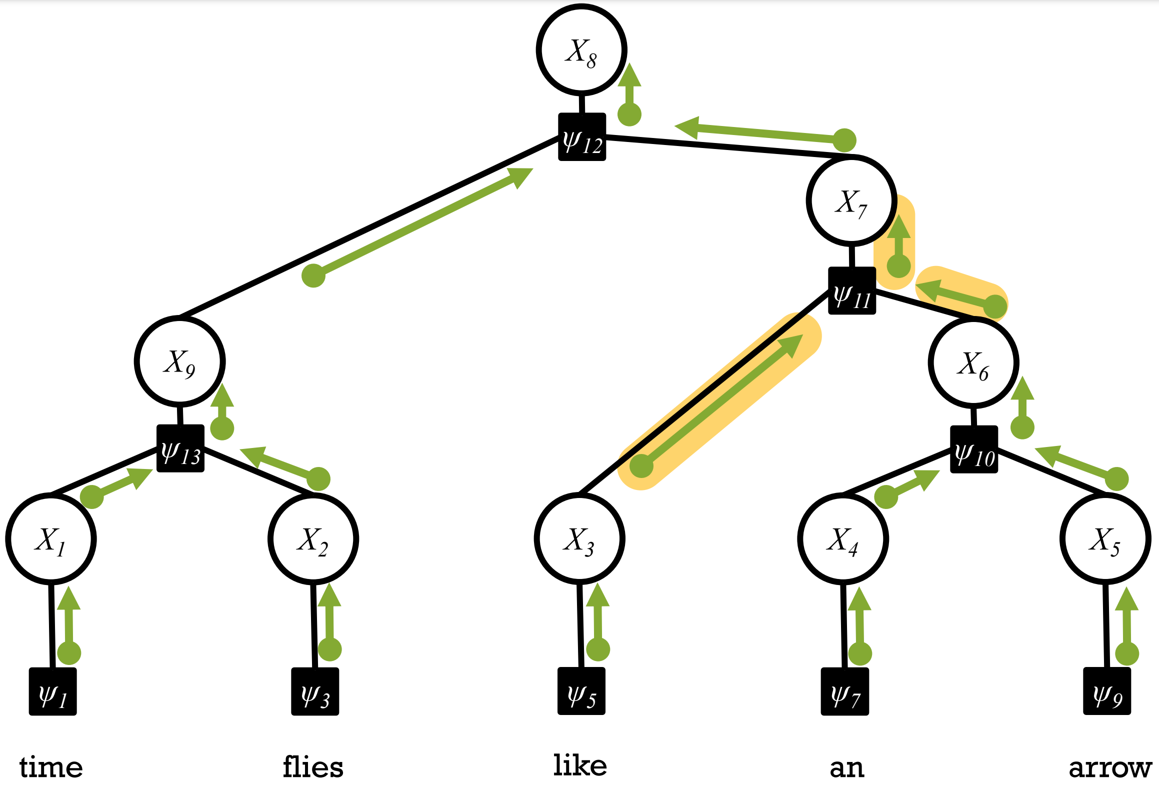
Slides adapted from Matt Gormley (2016)

(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.



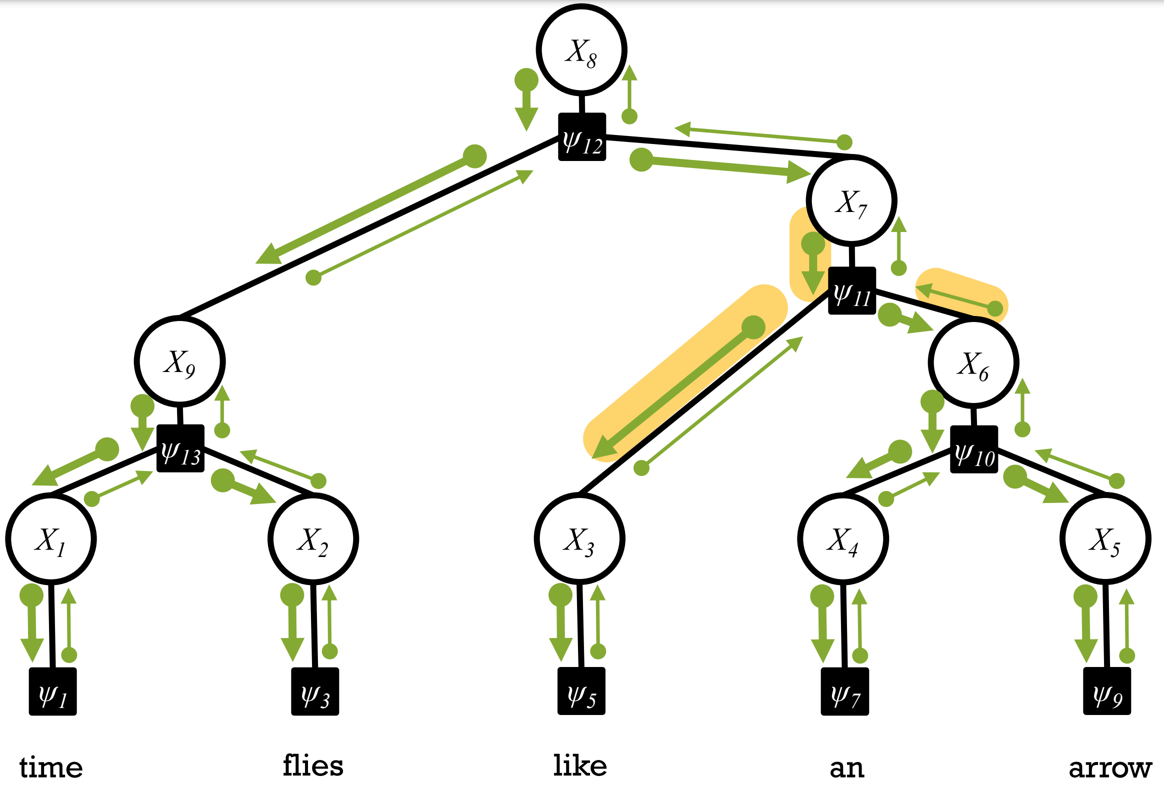
Slides adapted from Matt Gormley (2016)

(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.



Slides adapted from Matt Gormley (2016)

A note on the implementation

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

Variable to Factor message

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

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

Factor to Variable message

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

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

How about other queries?
(MPA, Evidence)

Example

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

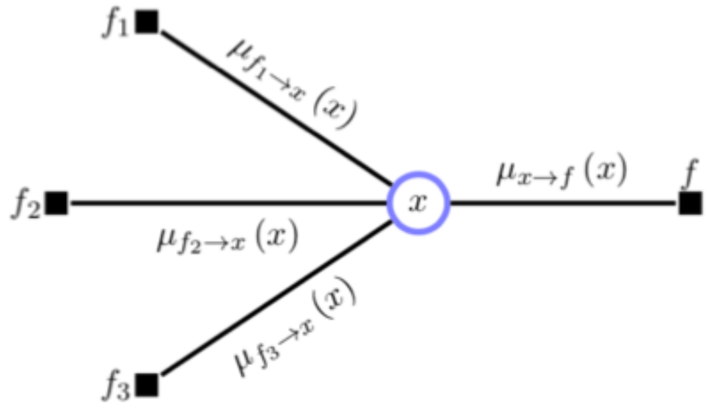
$$\underbrace{\max_{x_3} \phi(x_2,x_3)\gamma_4(x_3)}_{\gamma_3(x_2)}$$

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

The Max Product Algorithm

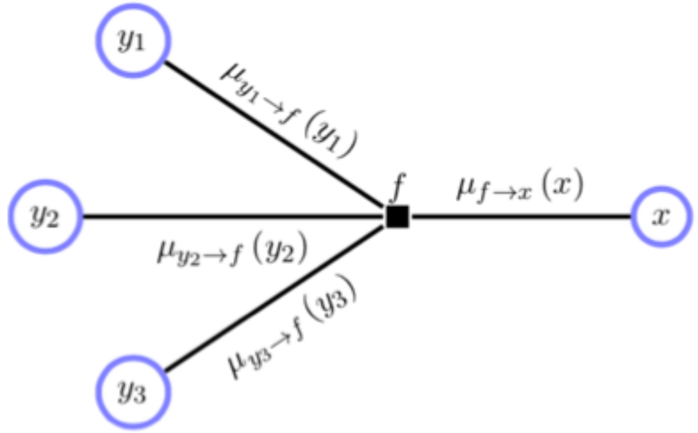
Variable to Factor message

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



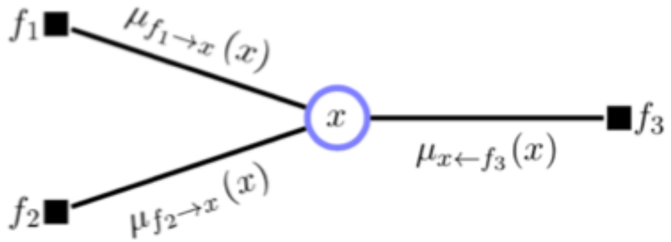
Factor to Variable message

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



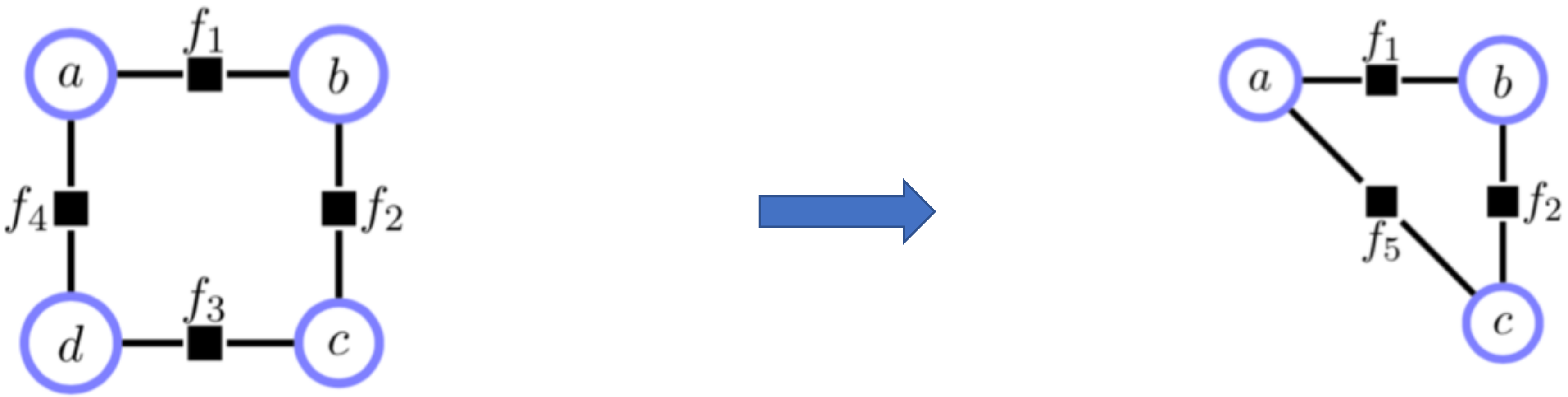
Maximal State

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



Can I use BP in a multiply connected graph?

Loops the trouble makers



$$p(a,b,c,d) = f_1(a,b)\, f_2(b,c)\, f_3(c,d)\, f_4(a,d)$$

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

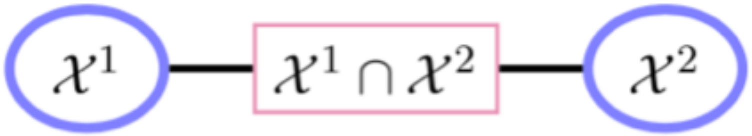
- One needs to account for the fact that the structure of the graph changes.
- The junction tree algorithm deals with this by combining variables to make a new singly connected graph for which the graph structure remains singly connected under variable elimination.

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 function

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

Example



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

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

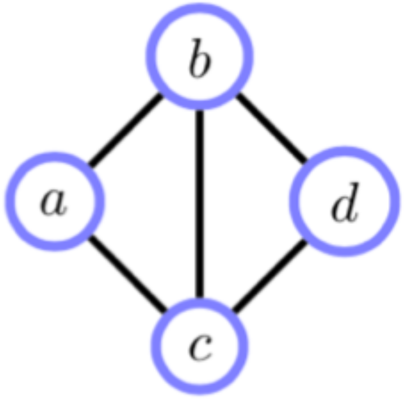
Don't confuse it with Factor Graph!

Example

$\mathcal{X}^1$ — $\mathcal{X}^1 \cap \mathcal{X}^2$ — $\mathcal{X}^2$

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

Example: probability density

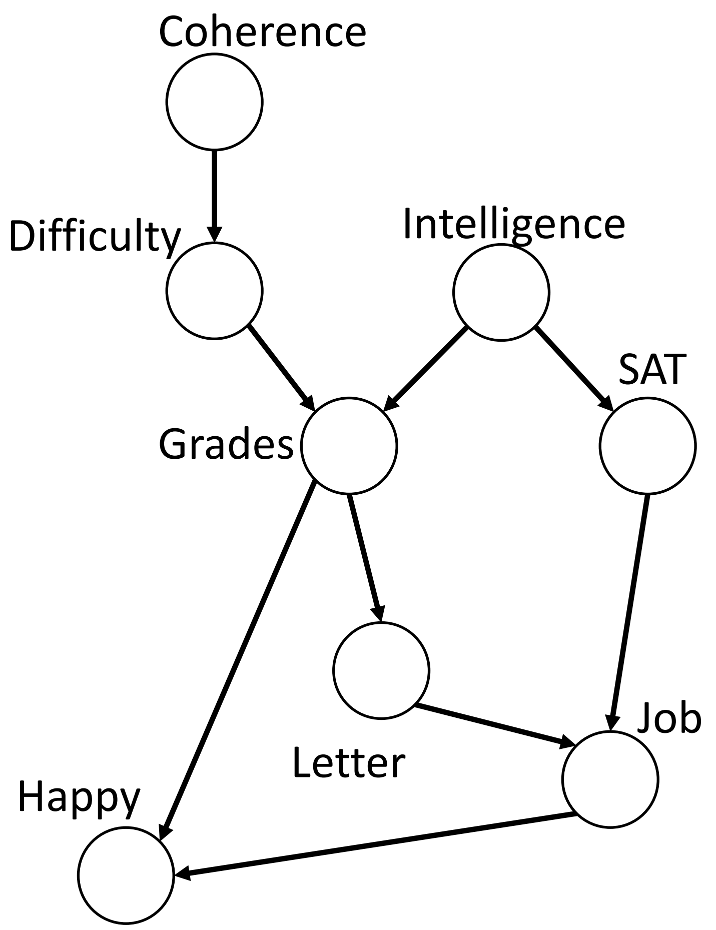


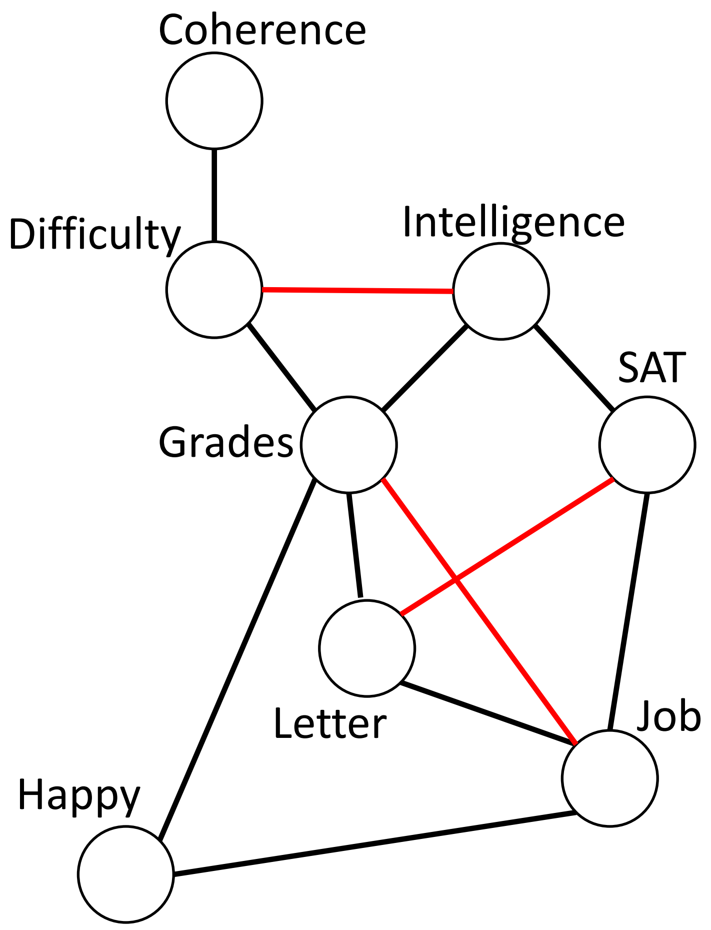
$$p(a,b,c,d) = \frac{\phi(a,b,c)\phi(b,c,d)}{Z} = \frac{p(a,b,c)p(b,c,d)}{p(c,b)}$$

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.

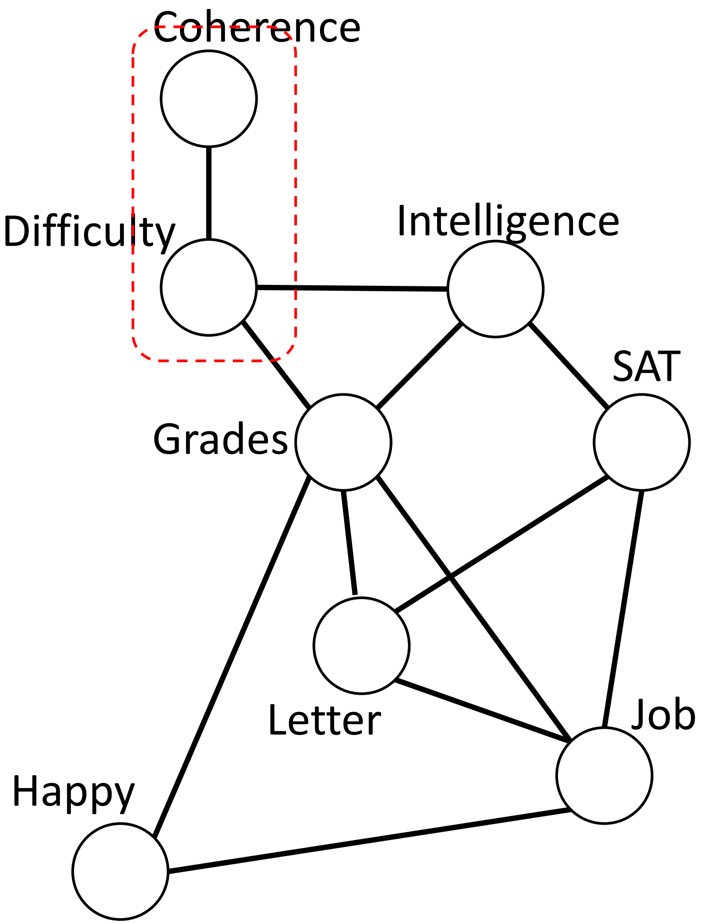
Let’s learn by an example….





Moralization

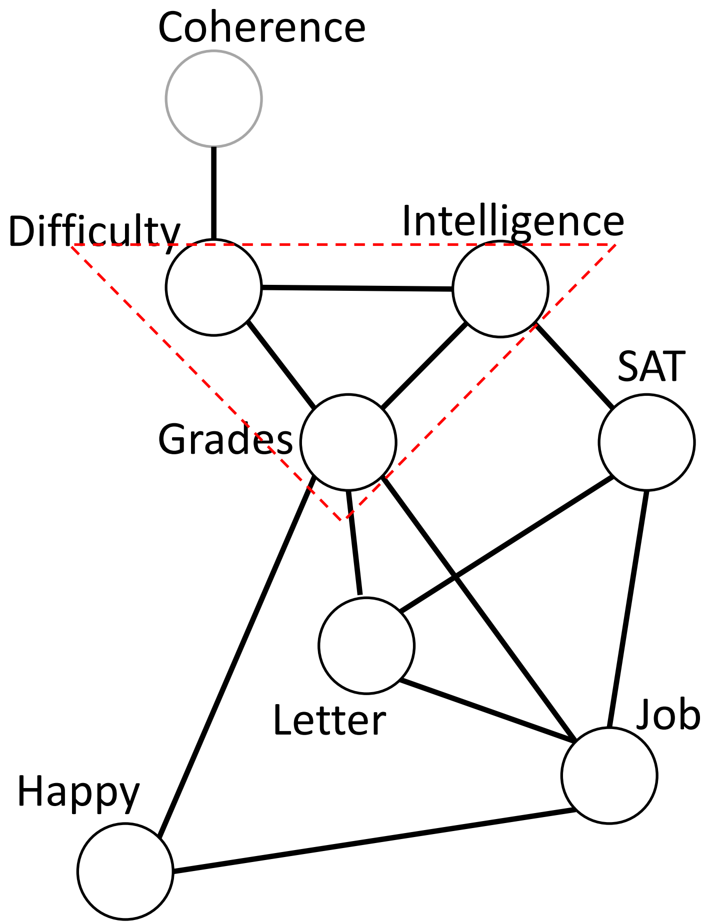
Let’s learn by an example….



Let’s pick an ordering for the variable elimination

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

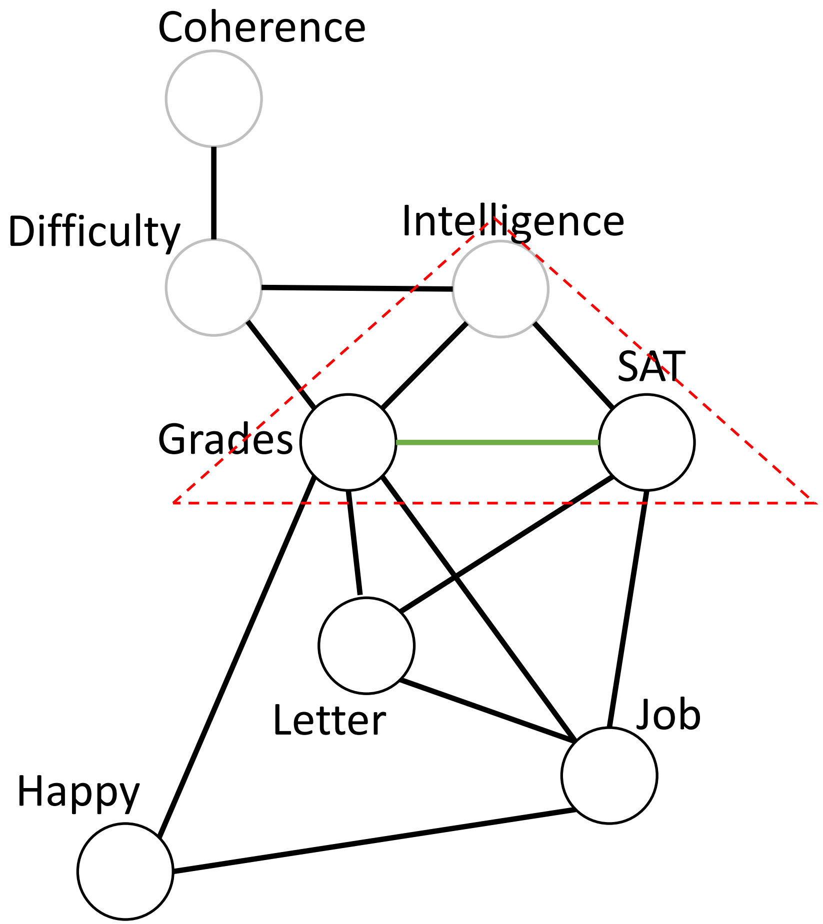
Let’s learn by an example....



Let’s pick an ordering for the variable elimination

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

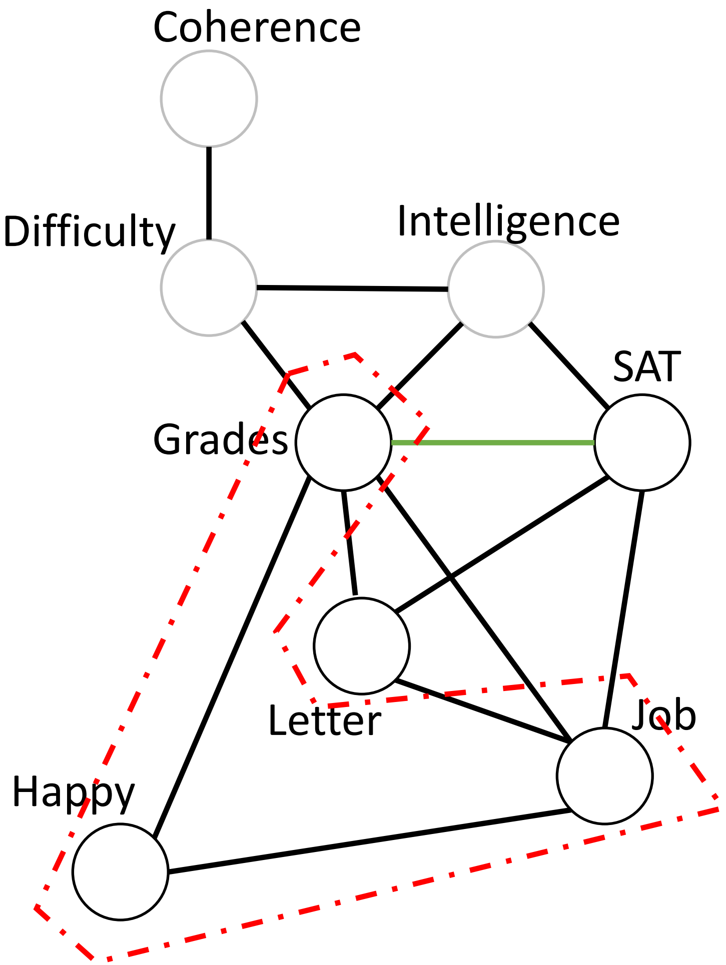
Let’s learn by an example....



Let’s pick an ordering for the variable elimination

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

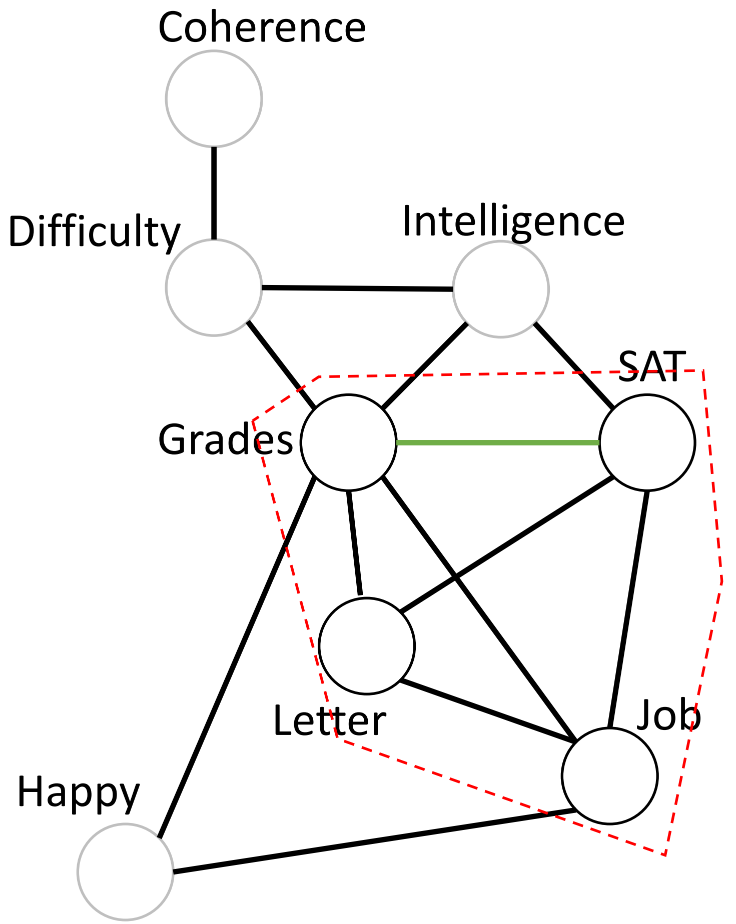
Let’s learn by an example….



Let’s pick an ordering for the variable elimination

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

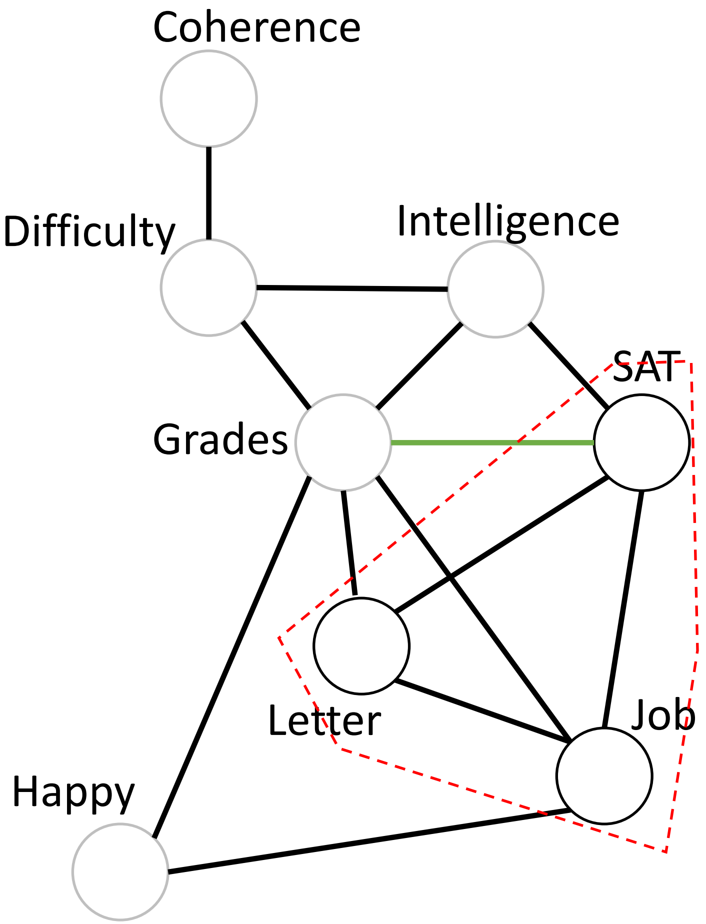
Let’s learn by an example....



Let’s pick an ordering for the variable elimination

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

Let’s learn by an example….

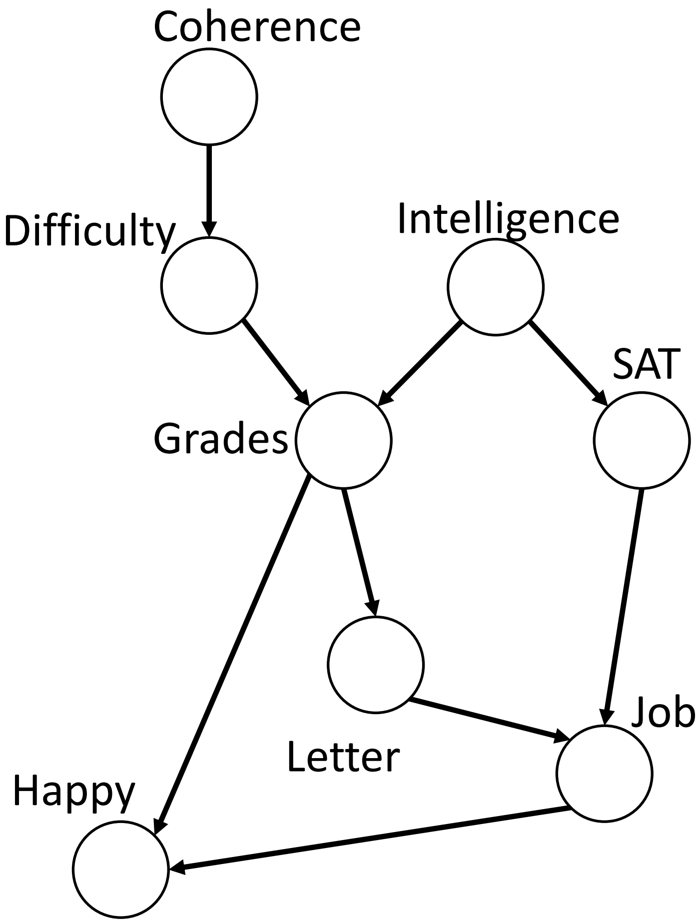


Let’s pick an ordering for the variable elimination

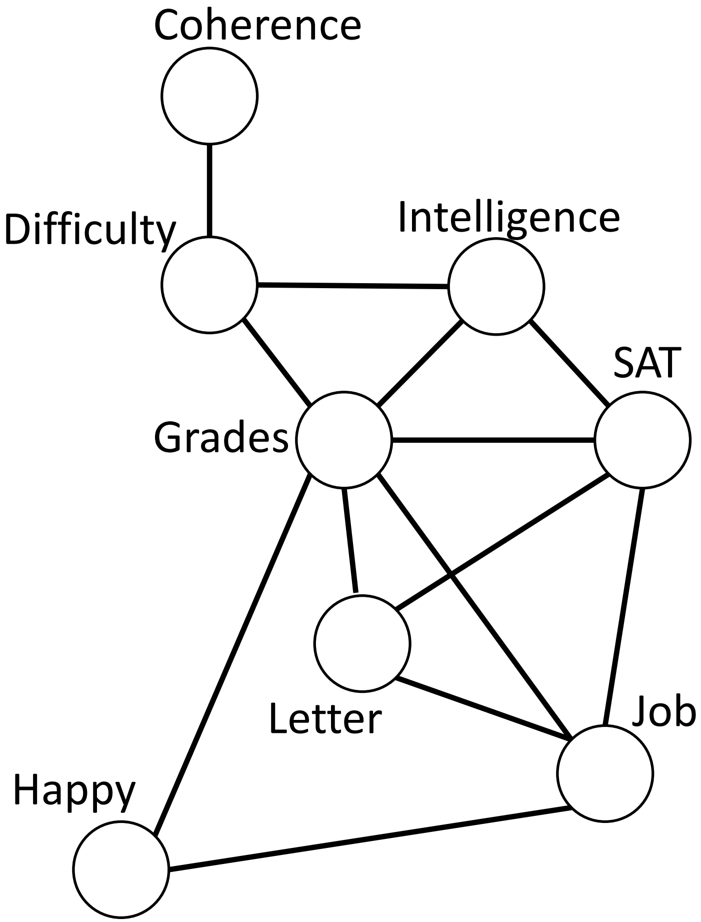
C, D, I, H, G, S, L

The rest is obvious

OK, what we got so far?



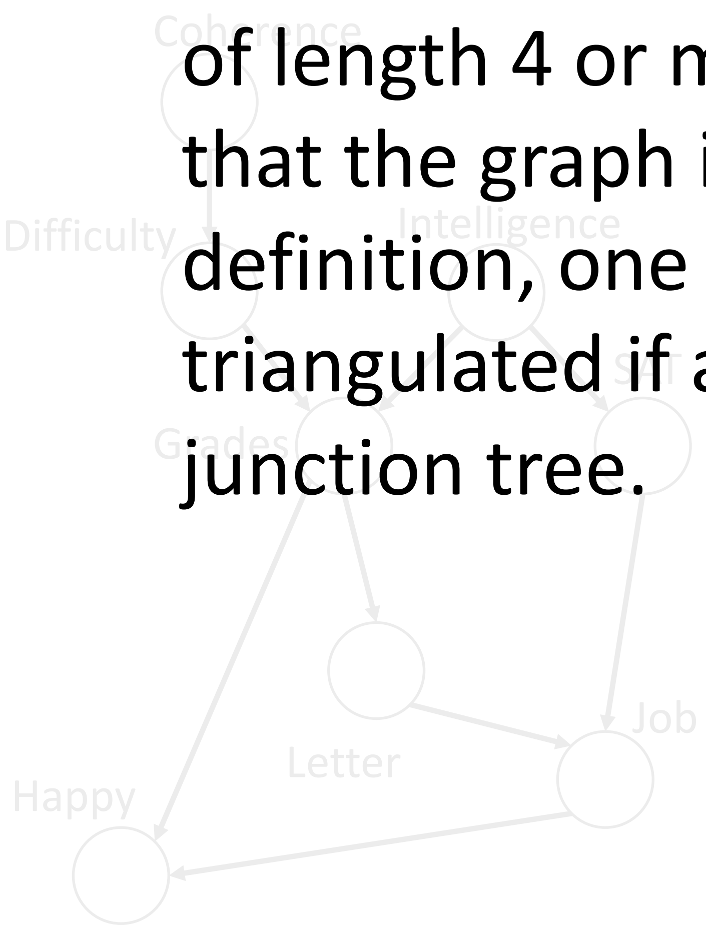
We started with



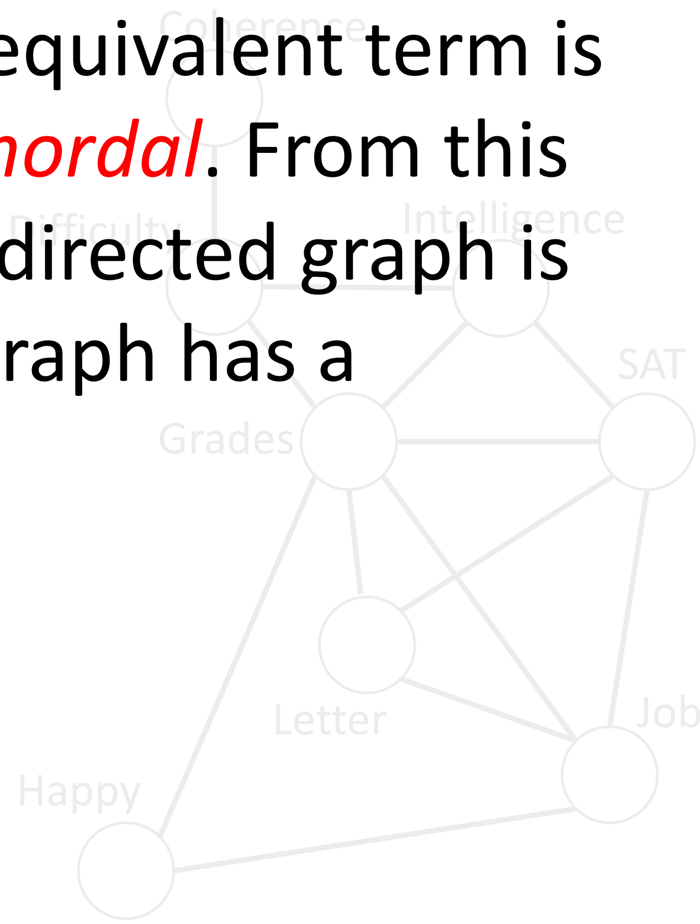
Moralized and Triangulated

OK, what we got so far?

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.

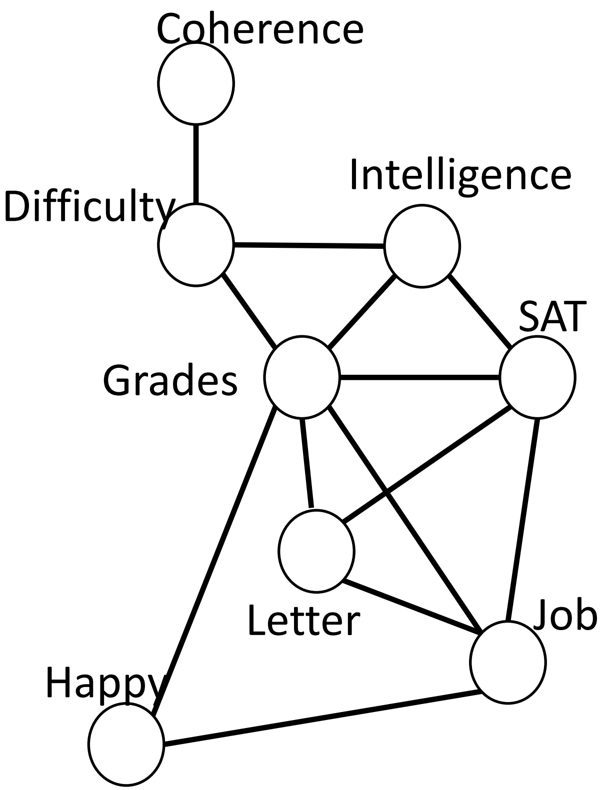


We started with

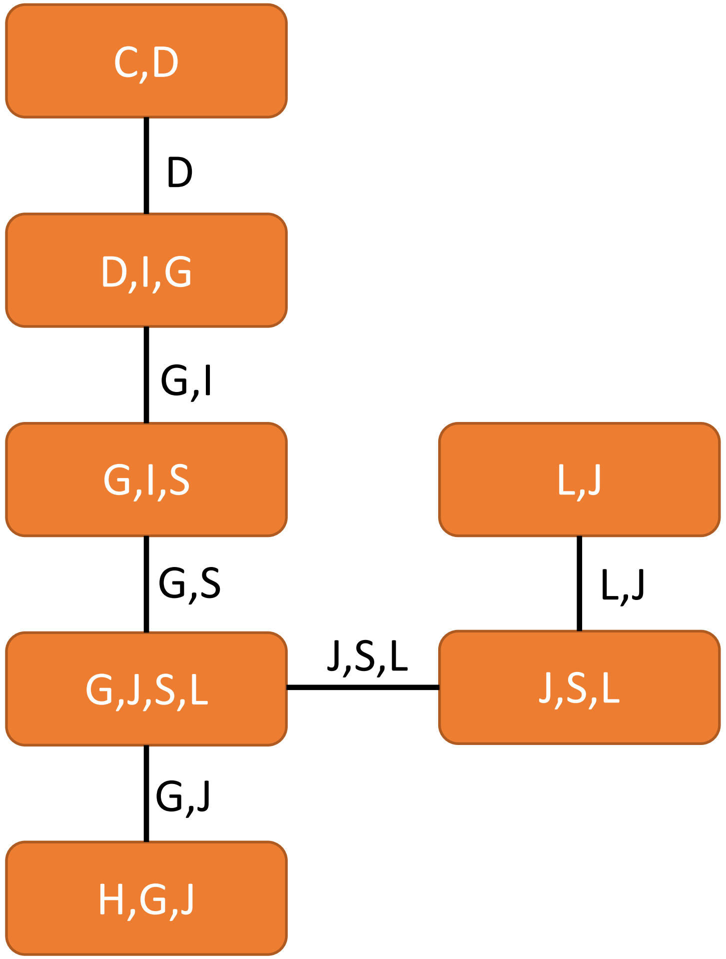


Moralized and Triangulated

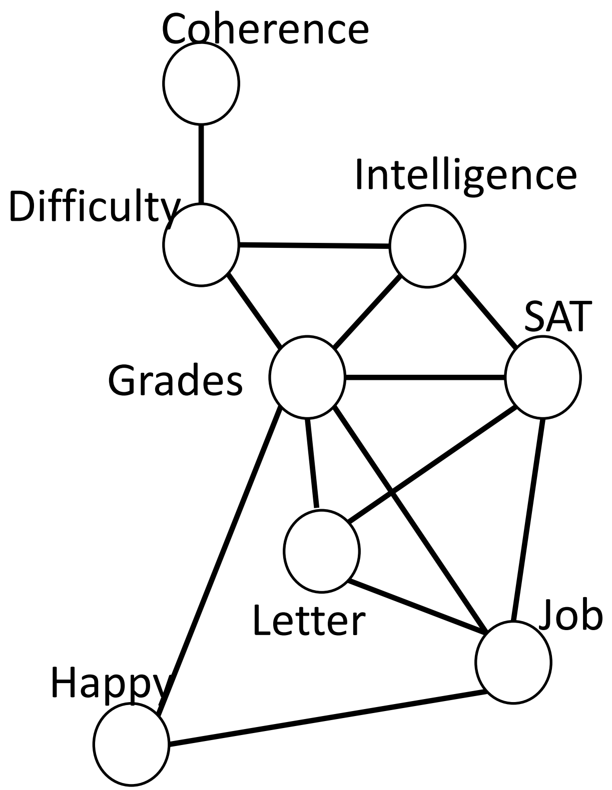
Let's build the Junction Tree



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

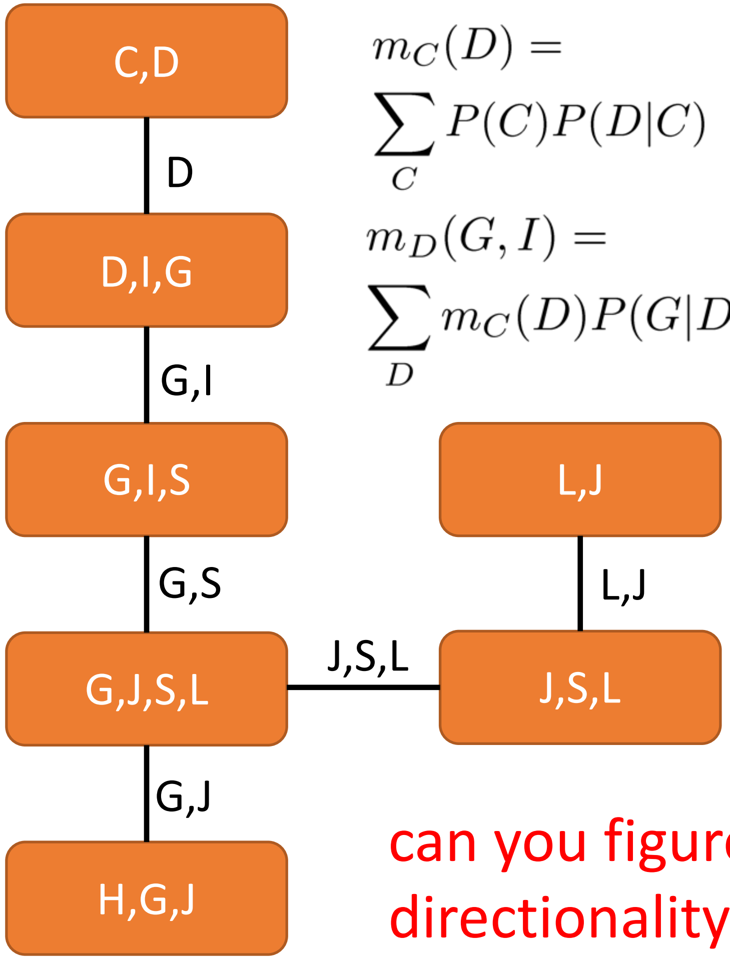


Pass the messages on the JT



The ordering

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



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

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

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

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

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

can you figure out the directionality?

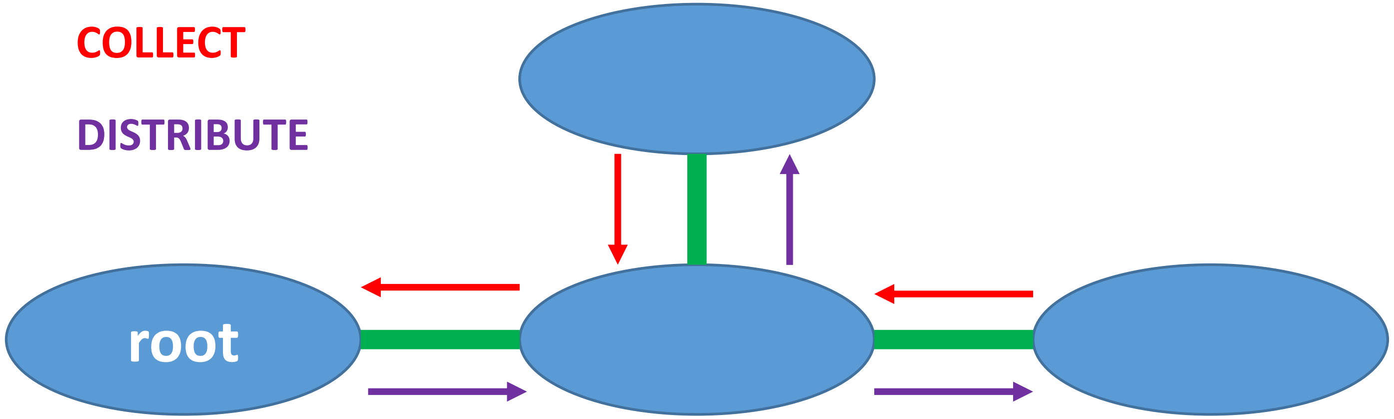
The message passing protocol

Message passing protocol

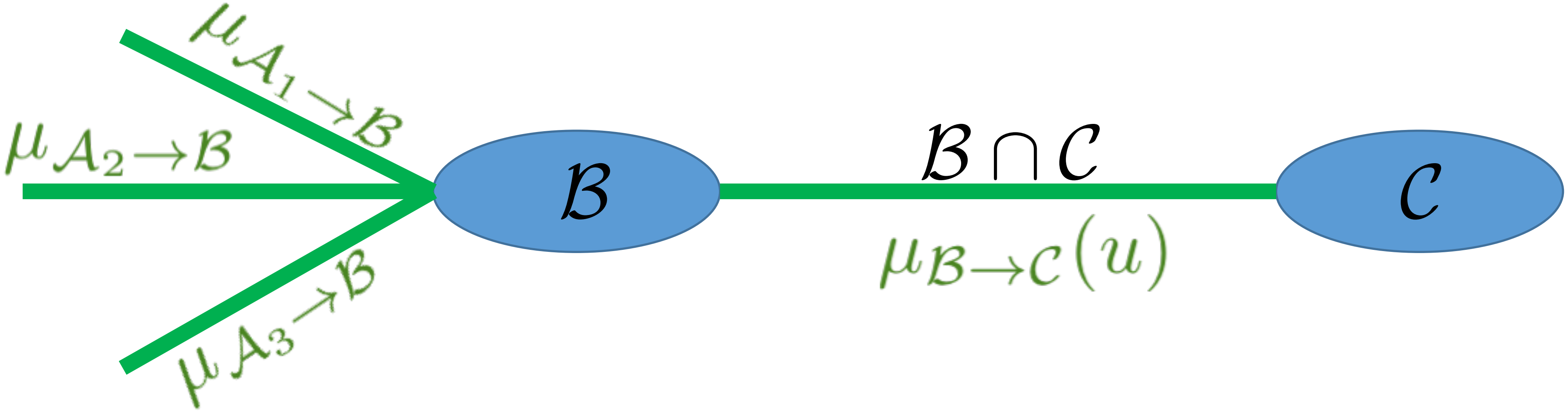
Cluster B is allowed to send a message to a neighbor C only after it has received messages from all neighbors except C.

```
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)
```



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



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

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

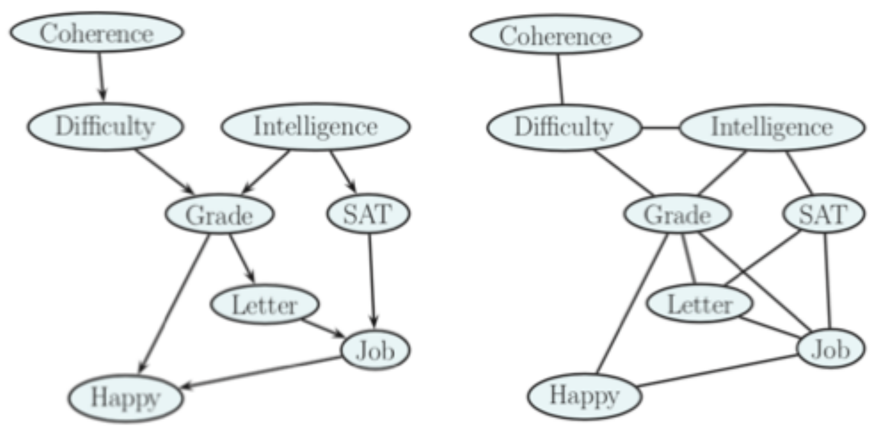
- BP is exact on trees.

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

- *If* it converges, convergence is fast near the fixed point.
- Many exciting applications:

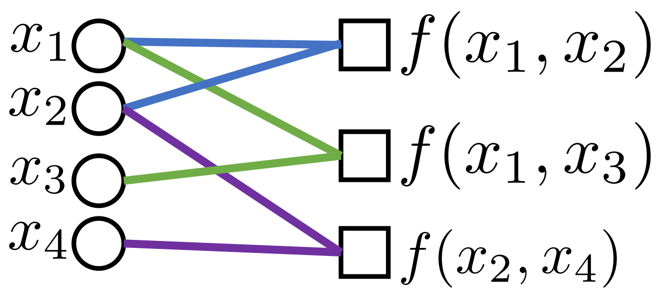
  - error correcting decoding (*MacKay, Yedidia, McEliece, Frey*)
  - vision (*Freeman, Weiss*)
  - bioinformatics (*Weiss*)
  - constraint satisfaction problems (*Dechter*)
  - game theory (*Kearns*)
  - …

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