



10-601B Introduction to Machine Learning

Factor Graphs and Exact Inference

Readings:

Bishop 8.1 and 8.2.2

Mitchell 6.11

Murphy 10

Matt Gormley

Lecture 22

November 13, 2016

Reminders

- Homework 6
 - due Mon., Nov. 21
- Final Exam
 - in-class Wed., Dec. 7

Exact Inference

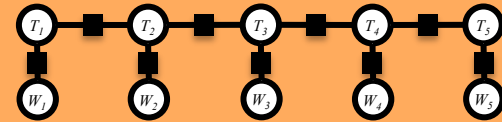
1. Data

$$\mathcal{D} = \{x^{(n)}\}_{n=1}^N$$

Sample 1:	<div>n time</div>	<div>v flies</div>	<div>p like</div>	<div>d an</div>	<div>n arrow</div>
Sample 2:	<div>n time</div>	<div>n flies</div>	<div>v like</div>	<div>d an</div>	<div>n arrow</div>
Sample 3:	<div>n flies</div>	<div>v fly</div>	<div>p with</div>	<div>n their</div>	<div>n wing</div>
Sample 4:	<div>p with</div>	<div>n time</div>	<div>n you</div>	<div>v will</div>	<div>v see</div>

2. Model

$$p(x | \theta) = \frac{1}{Z(\theta)} \prod_{C \in \mathcal{C}} \psi_C(x_C)$$



3. Objective

$$\ell(\theta; \mathcal{D}) = \sum_{n=1}^N \log p(x^{(n)} | \theta)$$

5. Inference

1. Marginal Inference

$$p(x_C) = \sum_{x': x'_C = x_C} p(x' | \theta)$$

2. Partition Function

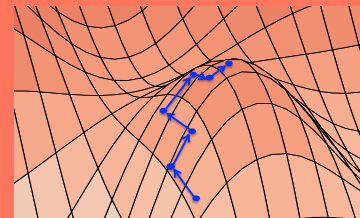
$$Z(\theta) = \sum_x \prod_{C \in \mathcal{C}} \psi_C(x_C)$$

3. MAP Inference

$$\hat{x} = \operatorname{argmax}_x p(x | \theta)$$

4. Learning

$$\theta^* = \operatorname{argmax}_{\theta} \ell(\theta; \mathcal{D})$$

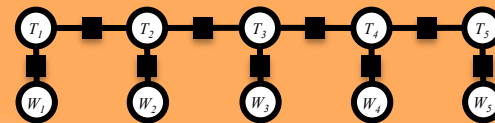


Goals for Today's Lecture

1. Unify MRFs and Bayes Nets with a new representation: **factor graphs**
2. Perform exact inference on factor graphs with two algorithms: **Variable Elimination** and **Sum-Product Belief Propagation**

Model

$$p(\mathbf{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C)$$



Objective

$$\ell(\boldsymbol{\theta}; \mathcal{D}) = \sum_{n=1}^N \log p(\mathbf{x}^{(n)} \mid \boldsymbol{\theta})$$

5. Inference

1. Marginal Inference

$$p(\mathbf{x}_C) = \sum_{\mathbf{x}': \mathbf{x}'_C = \mathbf{x}_C} p(\mathbf{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

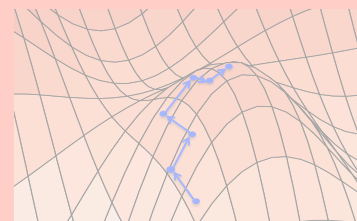
$$Z(\boldsymbol{\theta}) = \sum_{\mathbf{x}} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C)$$

3. MAP Inference

$$\hat{\mathbf{x}} = \operatorname{argmax}_{\mathbf{x}} p(\mathbf{x} \mid \boldsymbol{\theta})$$

4. Learning

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \mathcal{D})$$

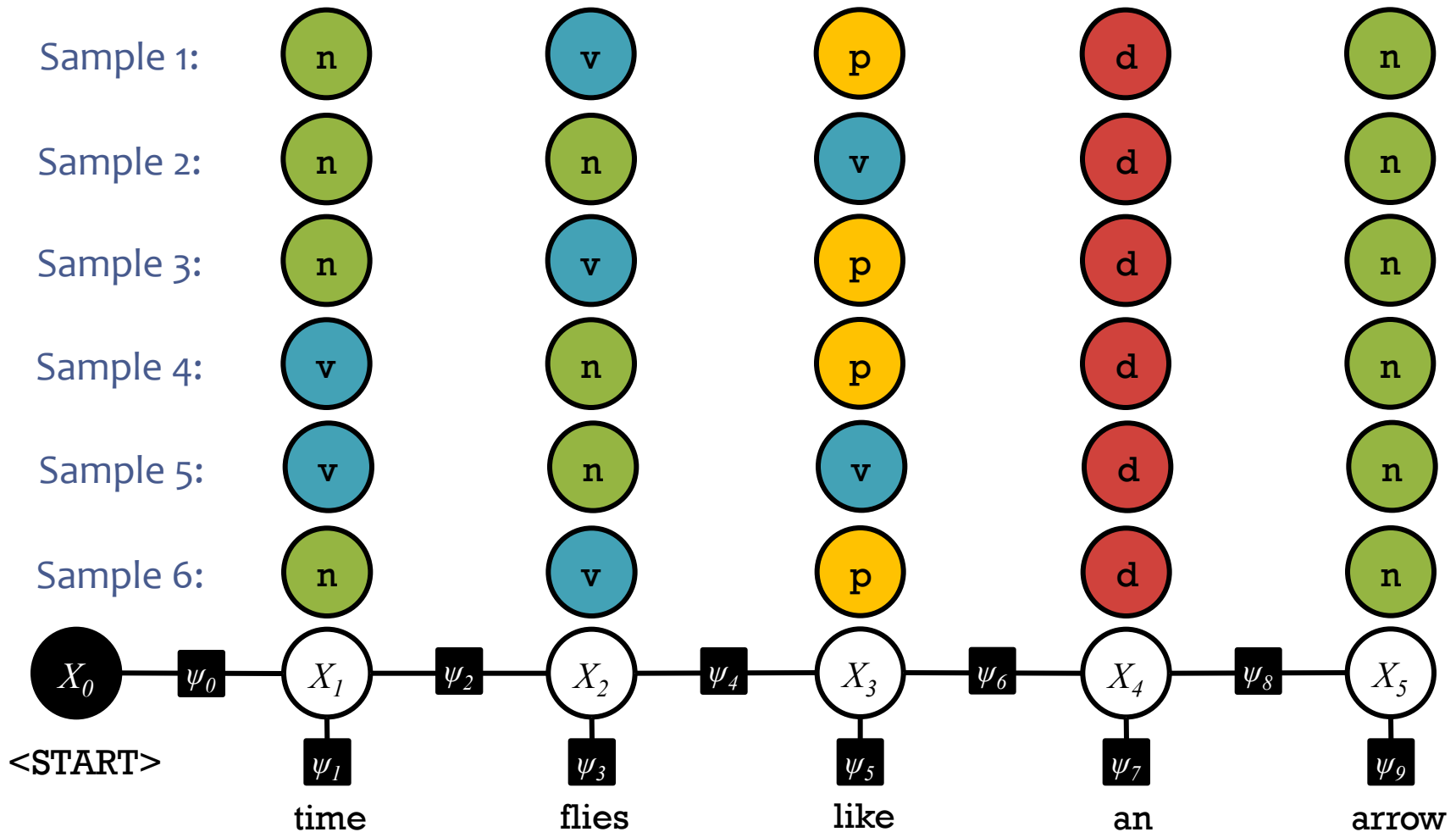


Representation of both directed and undirected graphical models

FACTOR GRAPHS

Sampling from a Joint Distribution

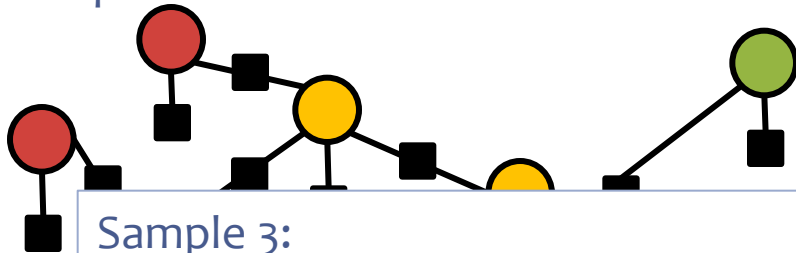
A **joint distribution** defines a probability $p(x)$ for each assignment of values x to variables X . This gives the **proportion** of samples that will equal x .



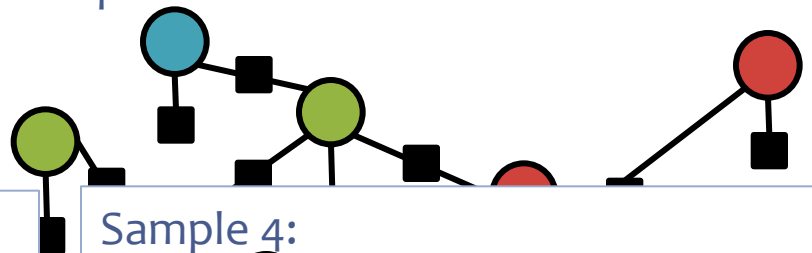
Sampling from a Joint Distribution

A **joint distribution** defines a probability $p(x)$ for each assignment of values x to variables X . This gives the **proportion** of samples that will equal x .

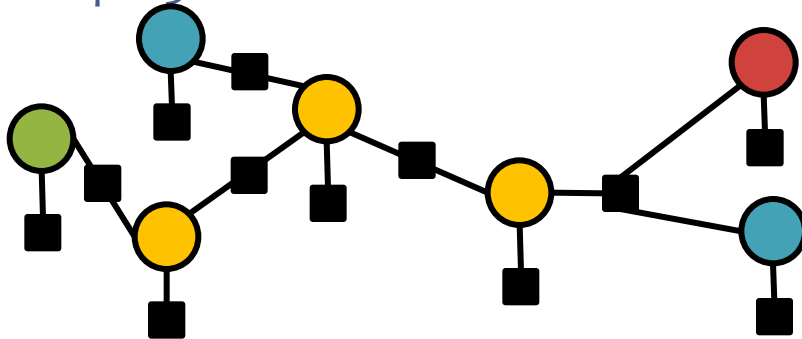
Sample 1:



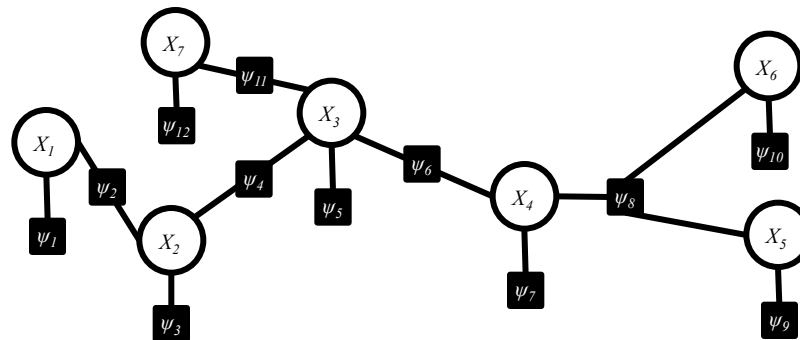
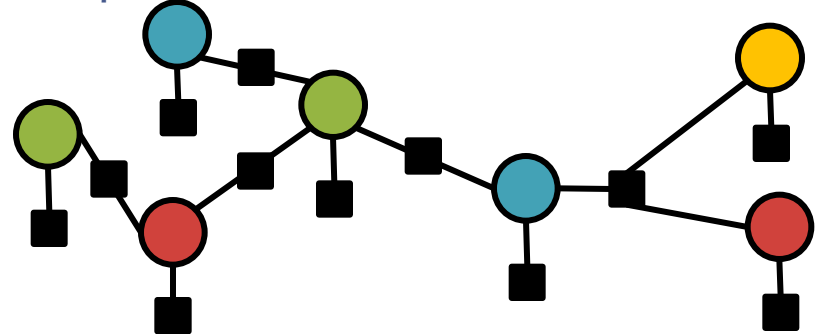
Sample 2:



Sample 3:











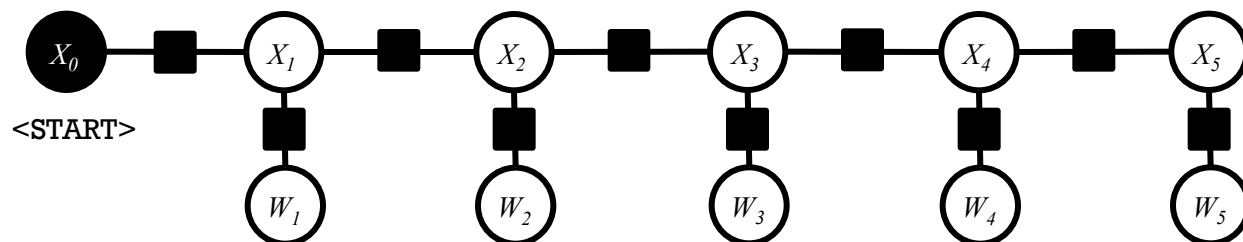
Sample 4:



Sampling from a Joint Distribution

A **joint distribution** defines a probability $p(x)$ for each assignment of values x to variables X . This gives the **proportion** of samples that will equal x .

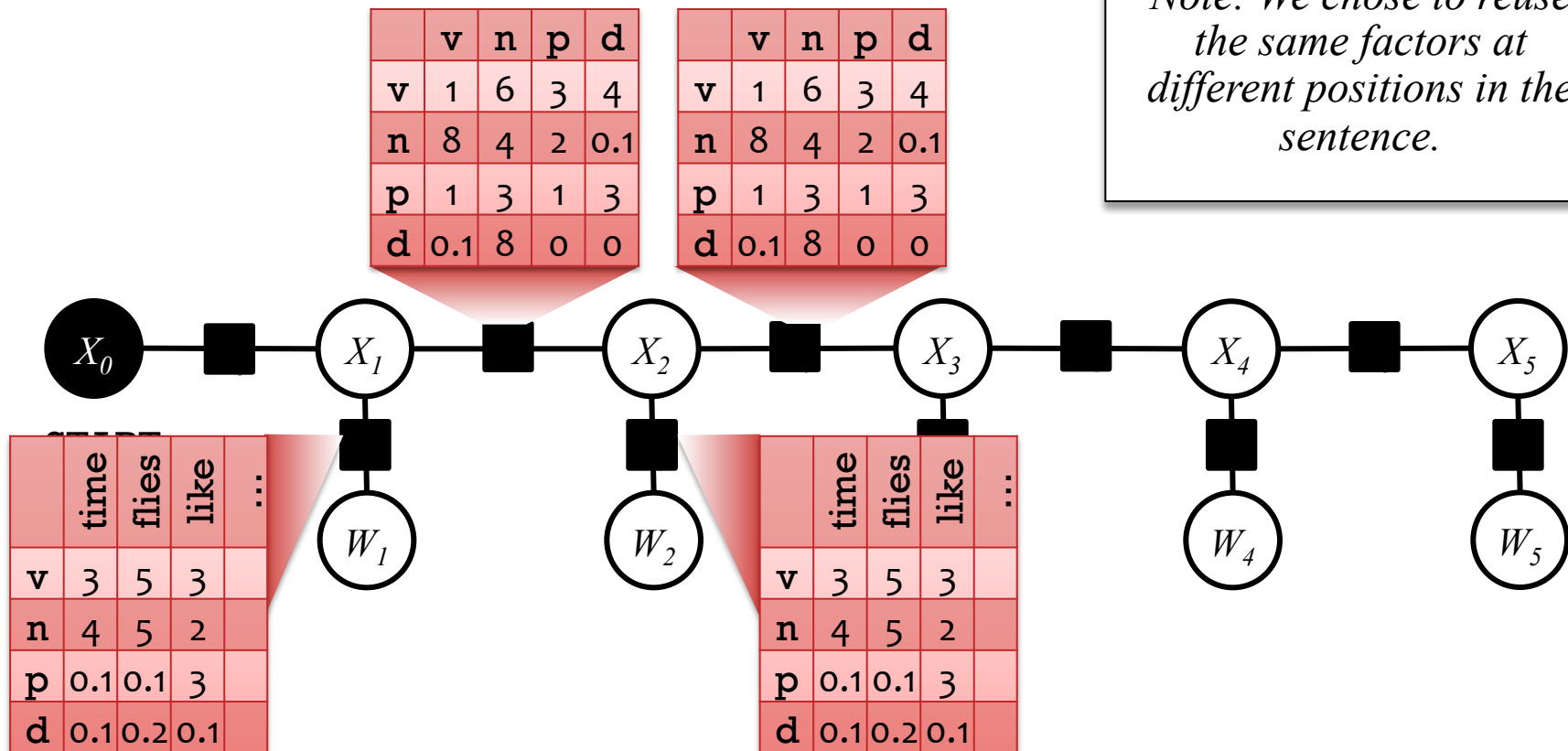
Sample 1:	 time	 flies	 like	 an	 arrow
Sample 2:	 time	 flies	 like	 an	 arrow
Sample 3:	 flies	 fly	 with	 their	 wings
Sample 4:	 with	 time	 you	 will	 see



Factors have local opinions (≥ 0)

Each black box looks at some of the tags X_i and words W_i

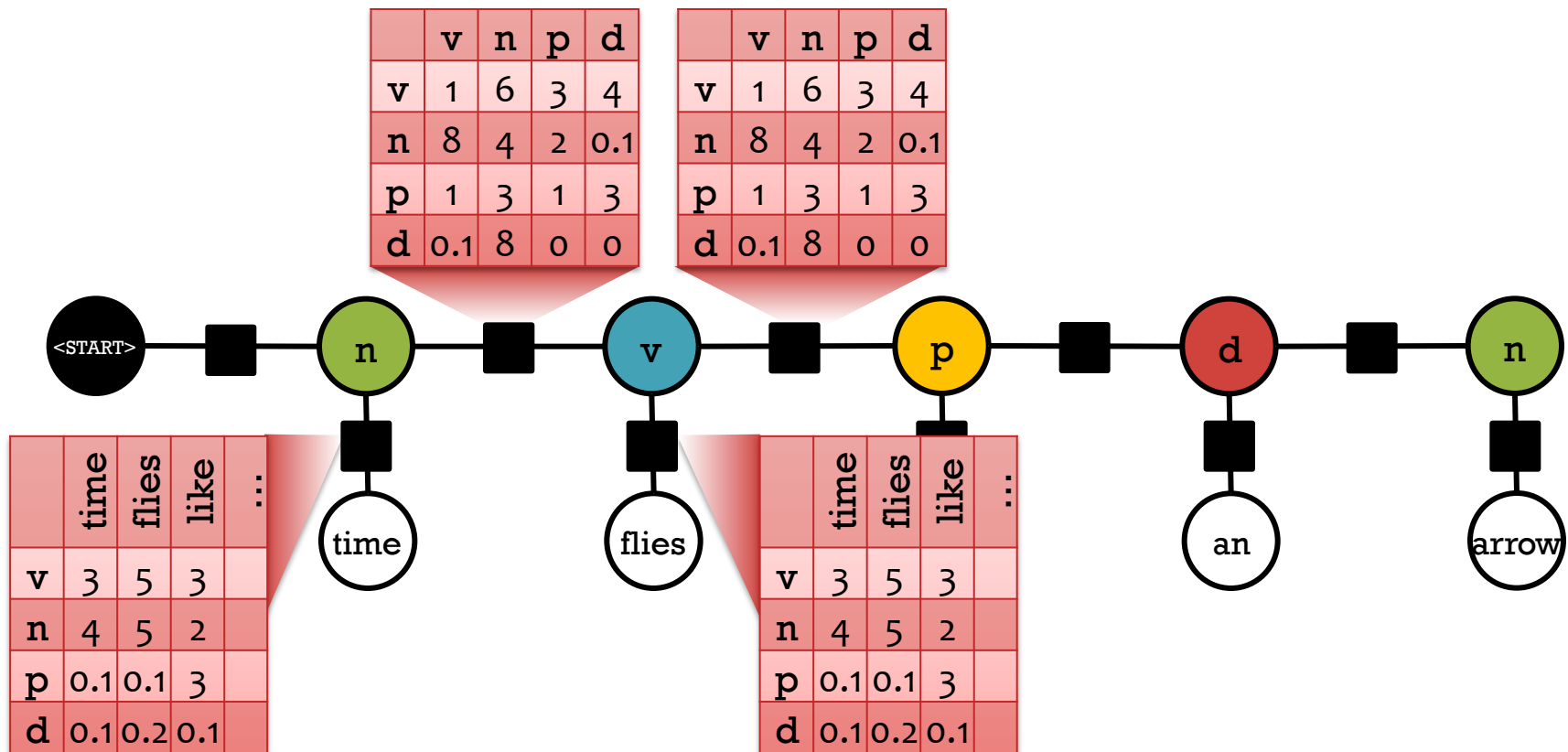
Note: We chose to reuse the same factors at different positions in the sentence.



Factors have local opinions (≥ 0)

Each black box looks at some of the tags X_i and words W_i

$$p(n, v, p, d, n, \text{time, flies, like, an, arrow}) = ?$$



Global probability = product of local opinions

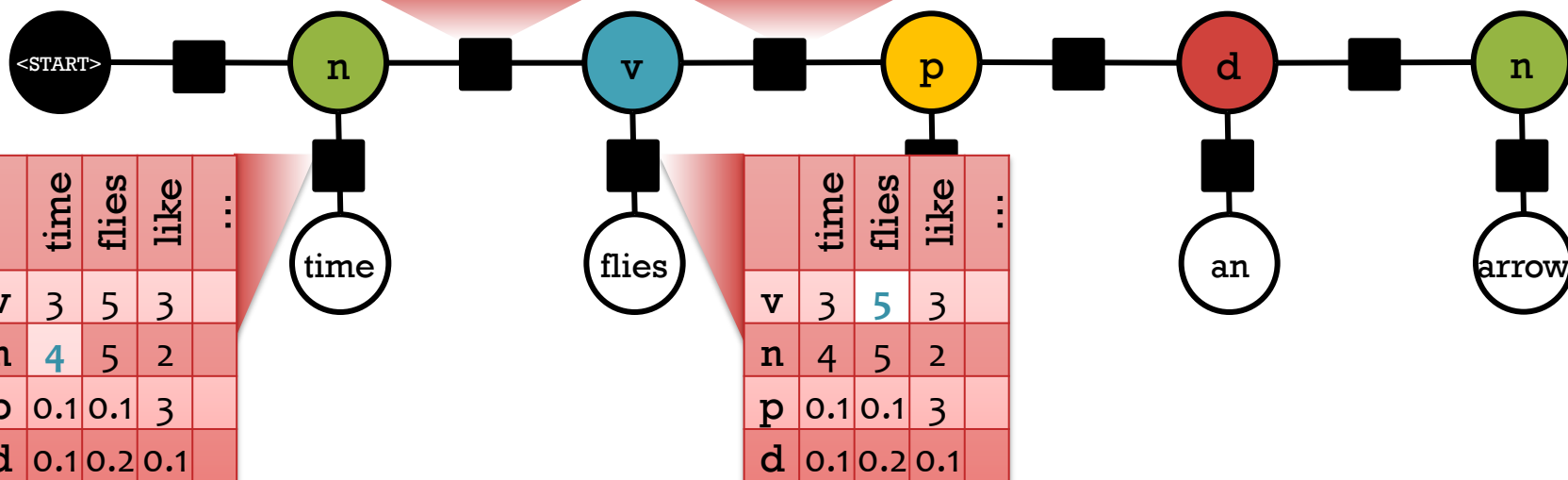
Each black box looks at some of the tags X_i and words W_i

$$p(n, v, p, d, n, \text{time, flies, like, an, arrow}) = \frac{1}{Z} (4 * 8 * 5 * 3 * \dots)$$

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

	v	n	p	d
v	1	6	3	4
n	8	4	2	0.1
p	1	3	1	3
d	0.1	8	0	0

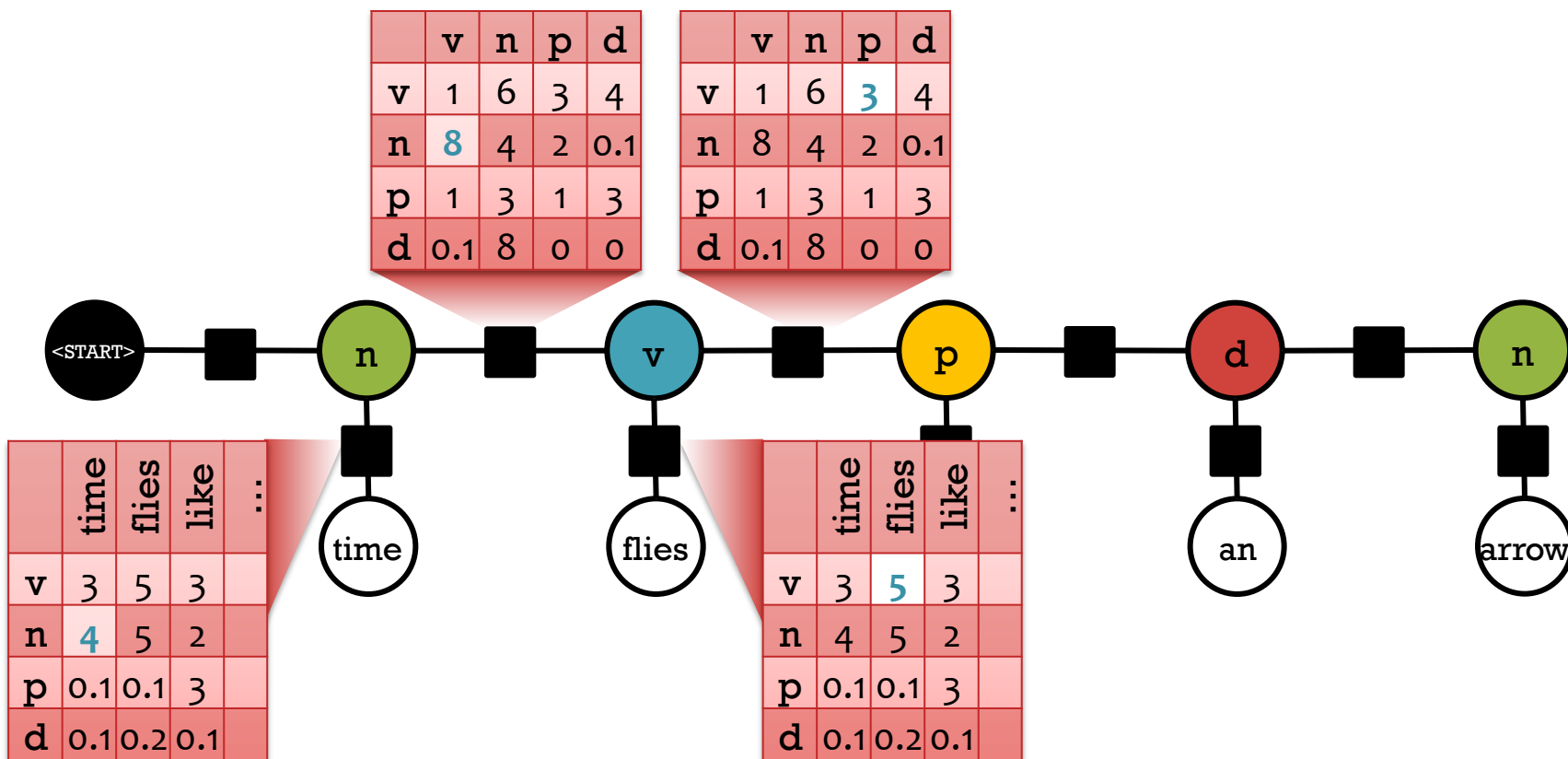
*Uh-oh! The probabilities of the various assignments sum up to $Z > 1$.
So divide them all by Z .*



Markov Random Field (MRF)

Joint distribution over tags X_i and words W_i
 The individual factors aren't *necessarily* probabilities.

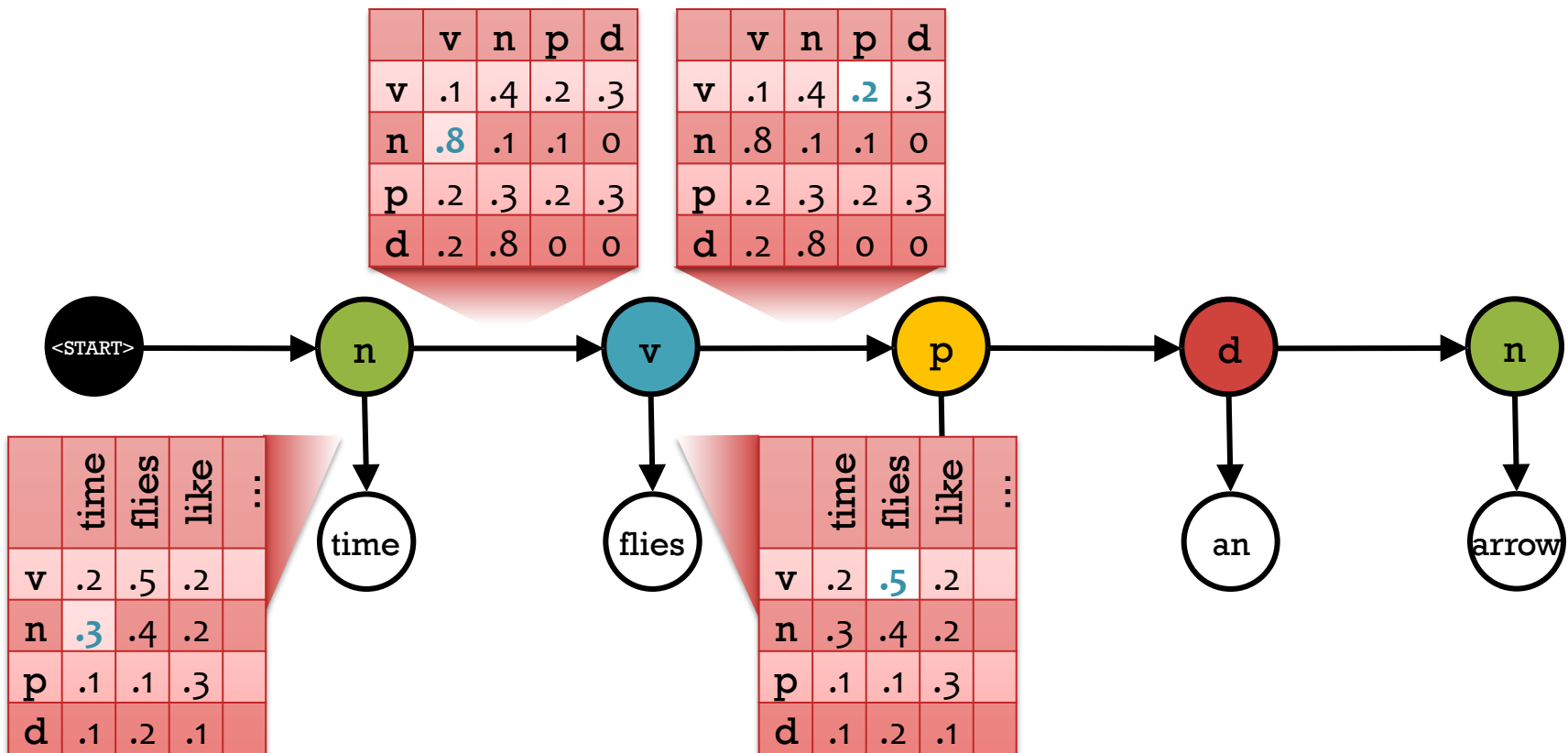
$$p(n, v, p, d, n, \text{time, flies, like, an, arrow}) = \frac{1}{Z} (4 * 8 * 5 * 3 * \dots)$$



Bayesian Networks

But sometimes we *choose* to make them probabilities. Constrain each row of a factor to sum to one. Now $Z = 1$.

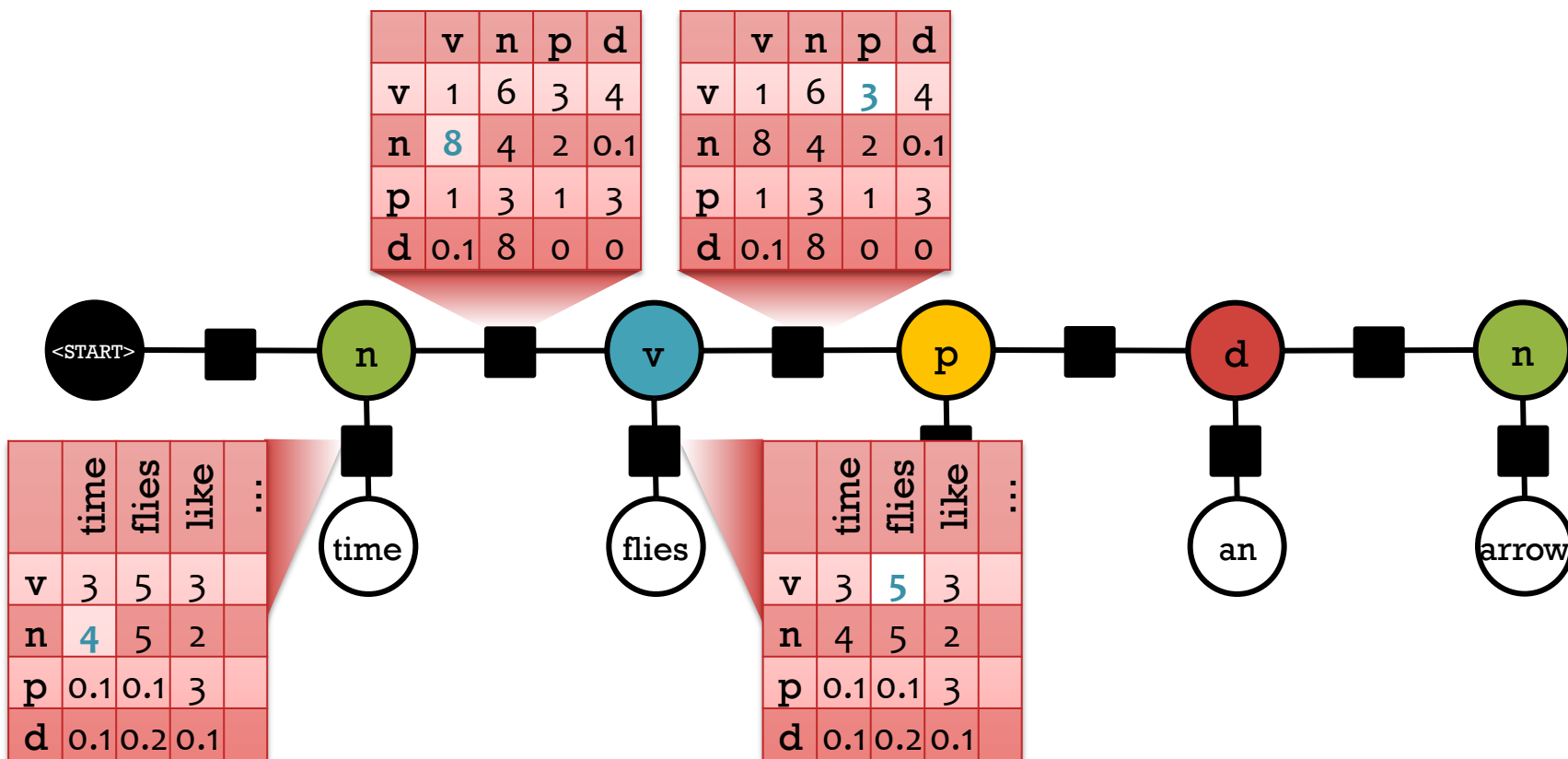
$$p(n, v, p, d, n, \text{time, flies, like, an, arrow}) = \cancel{\frac{1}{Z}} (.3 * .8 * .2 * .5 * \dots)$$



Markov Random Field (MRF)

Joint distribution over tags X_i and words W_i

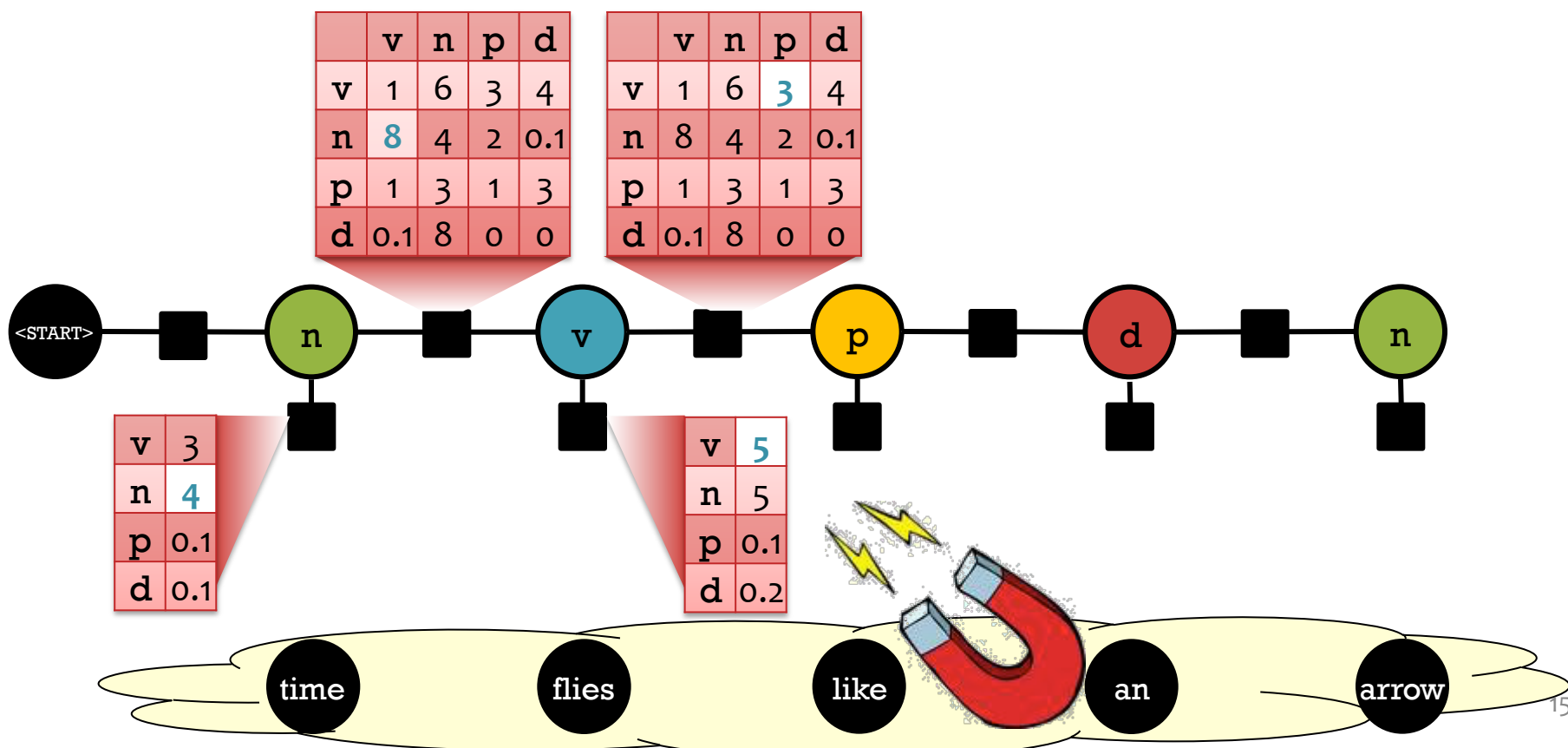
$$p(n, v, p, d, n, \text{time, flies, like, an, arrow}) = \frac{1}{Z} (4 * 8 * 5 * 3 * \dots)$$



Conditional Random Field (CRF)

Conditional distribution over tags X_i given words w_i .
The factors and Z are now specific to the sentence w .

$$p(n, v, p, d, n \mid \text{time, flies, like, an, arrow}) = \frac{1}{Z} (4 * 8 * 5 * 3 * \dots)$$



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:

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

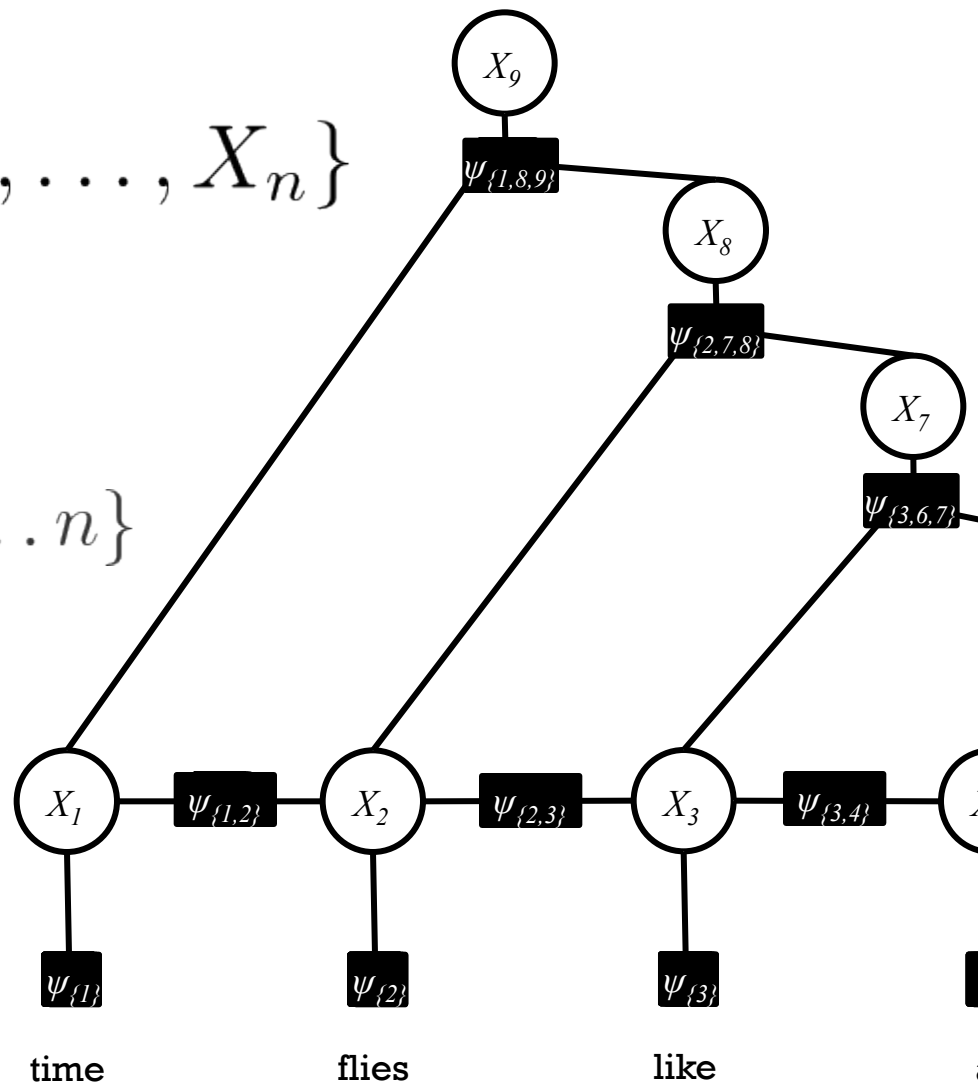
- Factors:

$$\psi_\alpha, \psi_\beta, \psi_\gamma, \dots$$

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

Joint Distribution

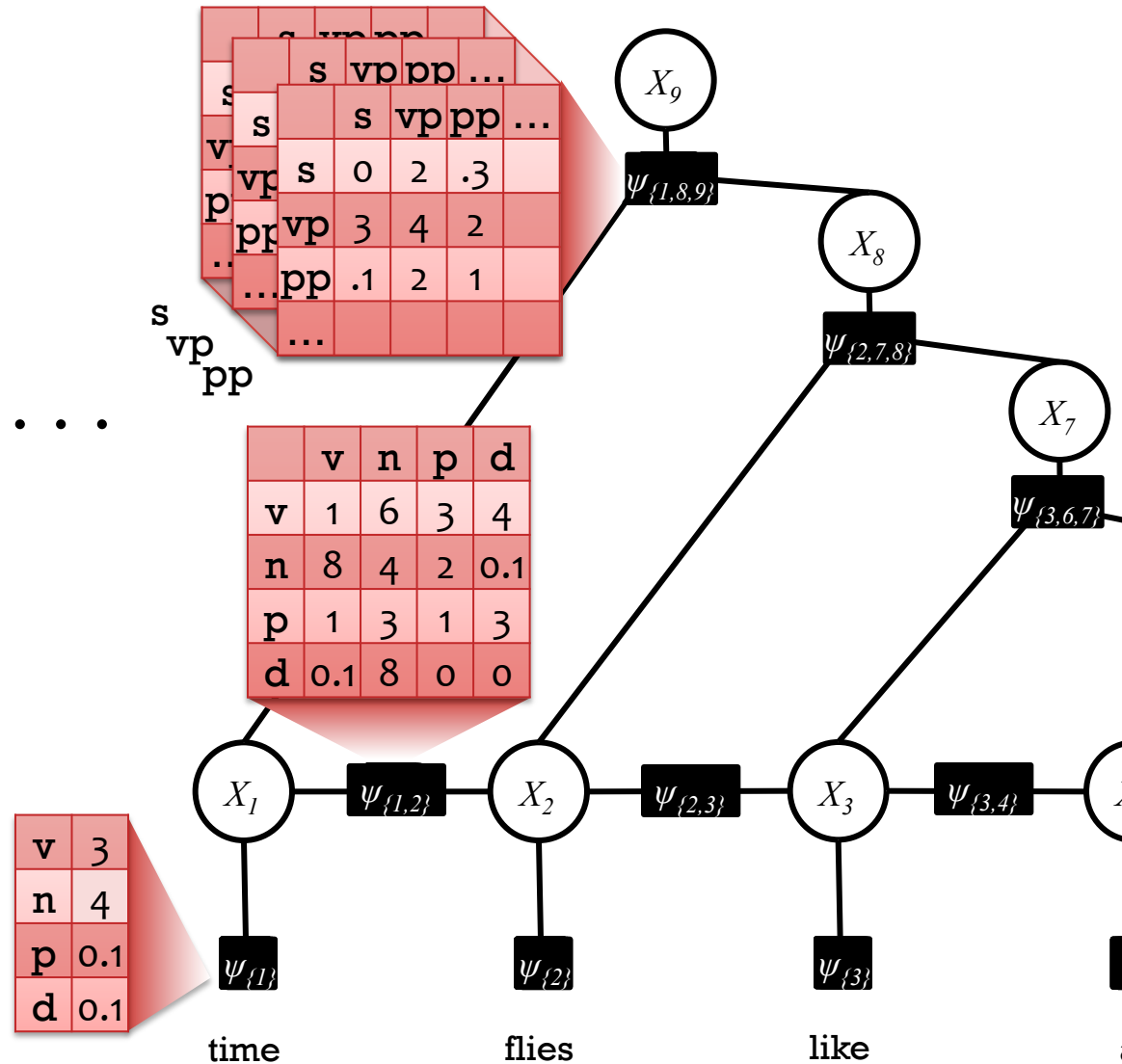
$$p(\mathbf{x}) = \frac{1}{Z} \prod_{\alpha} \psi_{\alpha}(\mathbf{x}_{\alpha})$$



Factors are Tensors

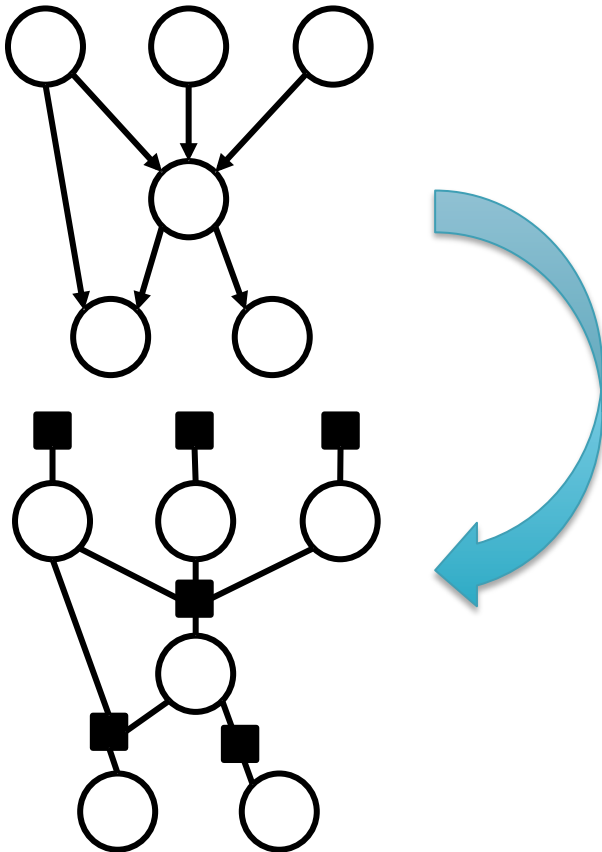
- Factors:

$\psi_\alpha, \psi_\beta, \psi_\gamma, \dots$

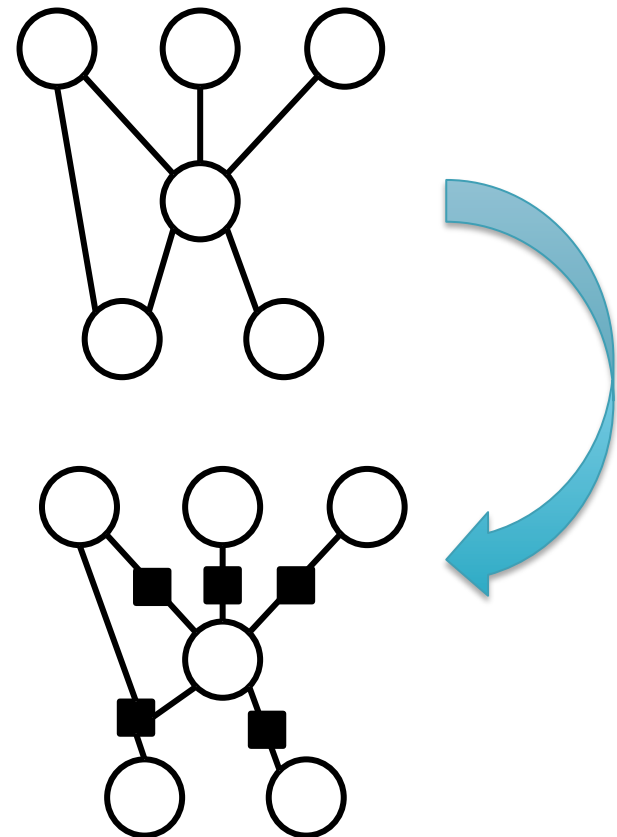


Converting to Factor Graphs

Each conditional and marginal distribution in a **directed GM** becomes a factor



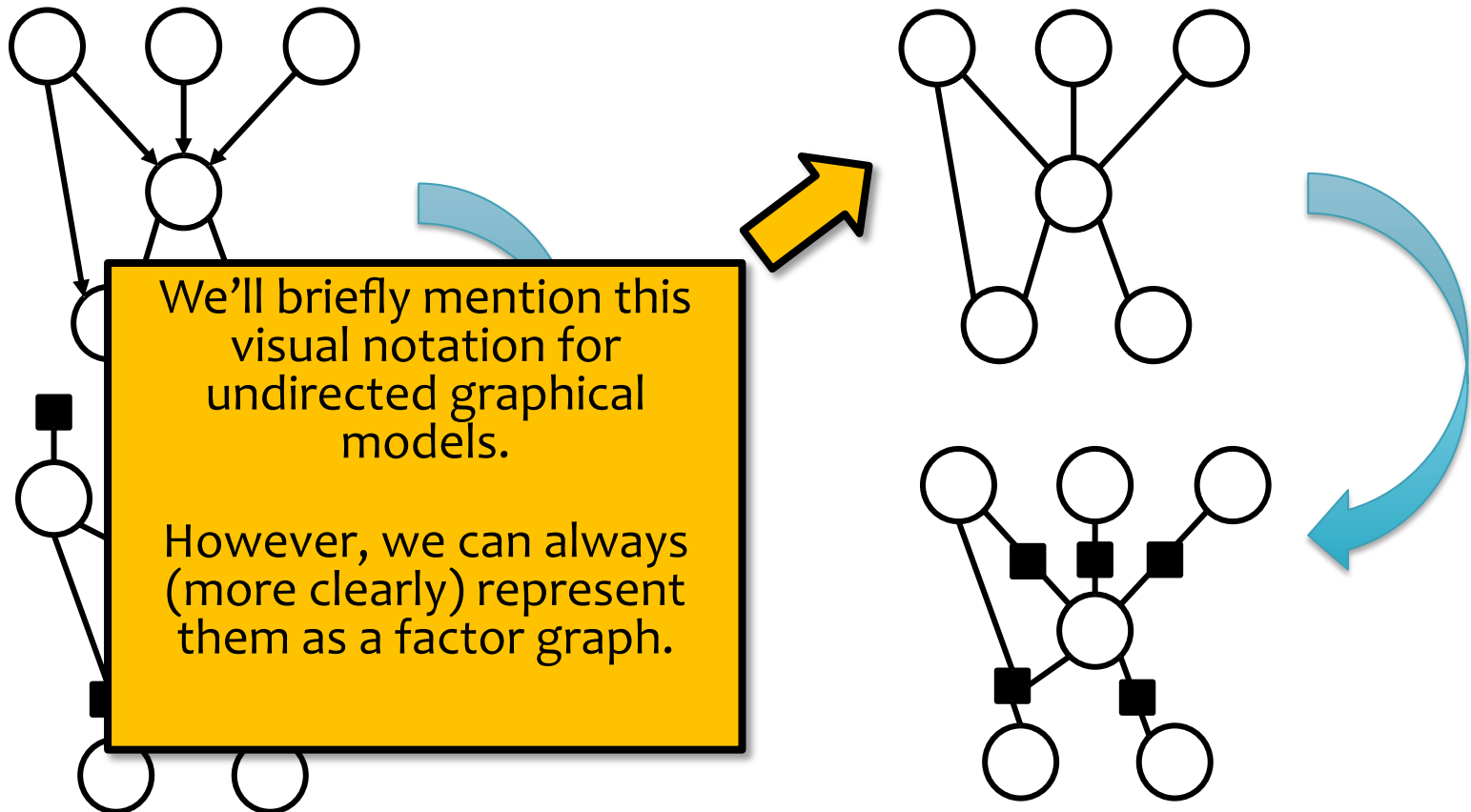
Each clique in an **undirected GM** becomes a factor



Converting to Factor Graphs

Each conditional and marginal distribution in a **directed GM** becomes a factor

Each clique in an **undirected GM** becomes a factor



Equivalence of directed and undirected trees

- Any undirected tree can be converted to a directed tree by choosing a root node and directing all edges away from it
- A directed tree and the corresponding undirected tree make the same conditional independence assertions
- Parameterizations are essentially the same.

– Undirected tree:

$$p(x) = \frac{1}{Z} \left(\prod_{i \in V} \psi(x_i) \prod_{(i,j) \in E} \psi(x_i, x_j) \right)$$

– Directed tree:

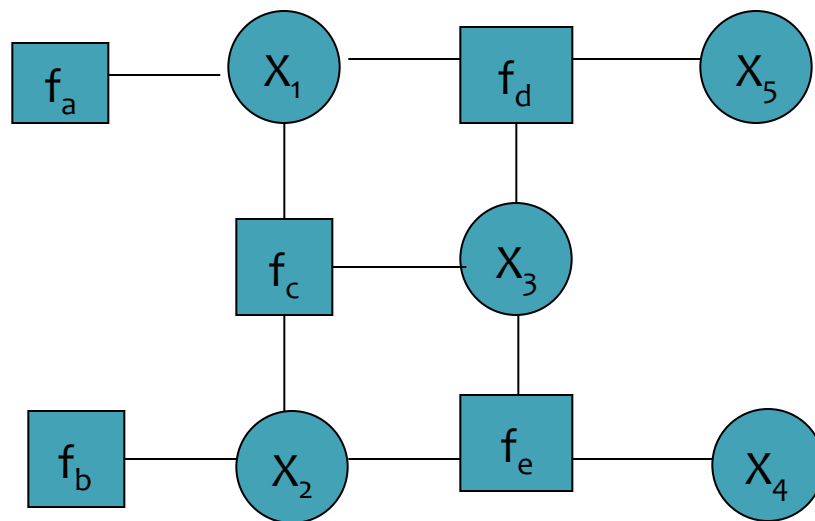
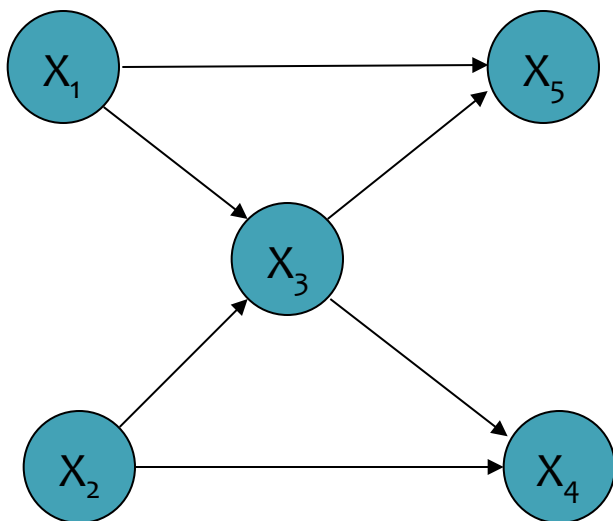
$$p(x) = p(x_r) \prod_{(i,j) \in E} p(x_j | x_i)$$

– Equivalence:

$$\begin{aligned} \psi(x_r) &= p(x_r); \quad \psi(x_i, x_j) = p(x_j | x_i); \\ Z &= 1, \quad \psi(x_i) = 1 \end{aligned}$$

Factor Graph Examples

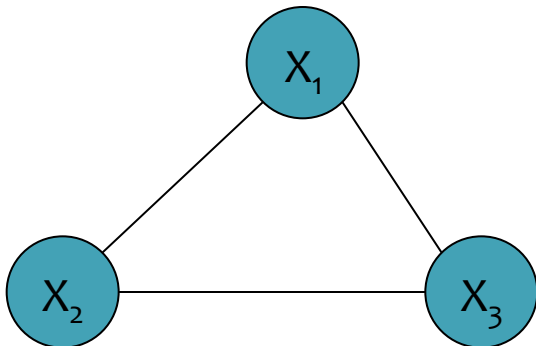
- Example 1



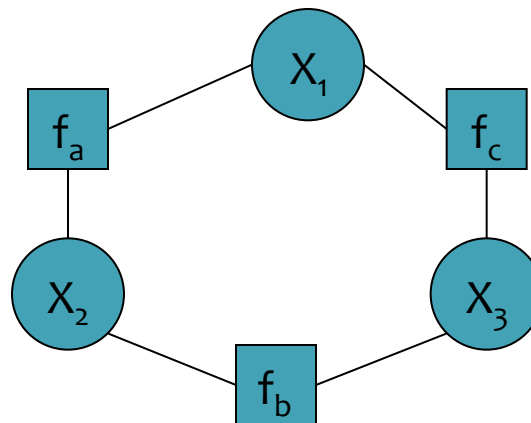
$P(X_1)$	$P(X_2)$	$P(X_3 X_1, X_2)$	$P(X_5 X_1, X_3)$	$P(X_4 X_2, X_3)$
↓	↓	↓	↓	↓
$f_a(X_1)$	$f_b(X_2)$	$f_c(X_3, X_1, X_2)$	$f_d(X_5, X_1, X_3)$	$f_e(X_4, X_2, X_3)$

Factor Graph Examples

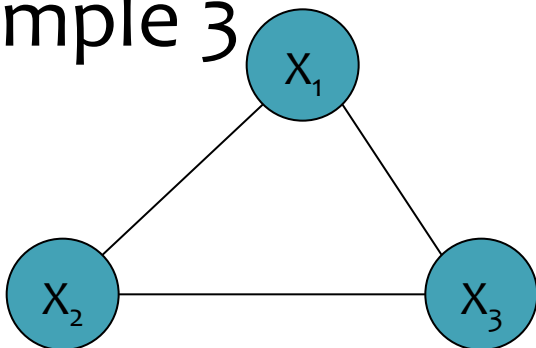
- Example 2



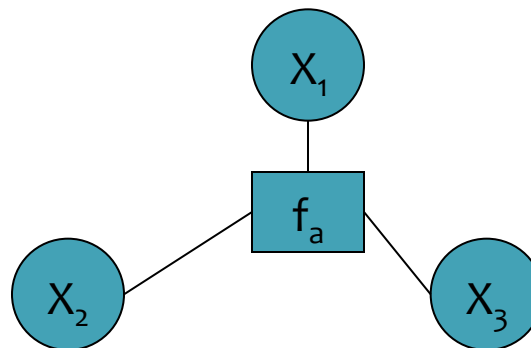
$$\psi(x_1, x_2, x_3) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_3, x_1)$$



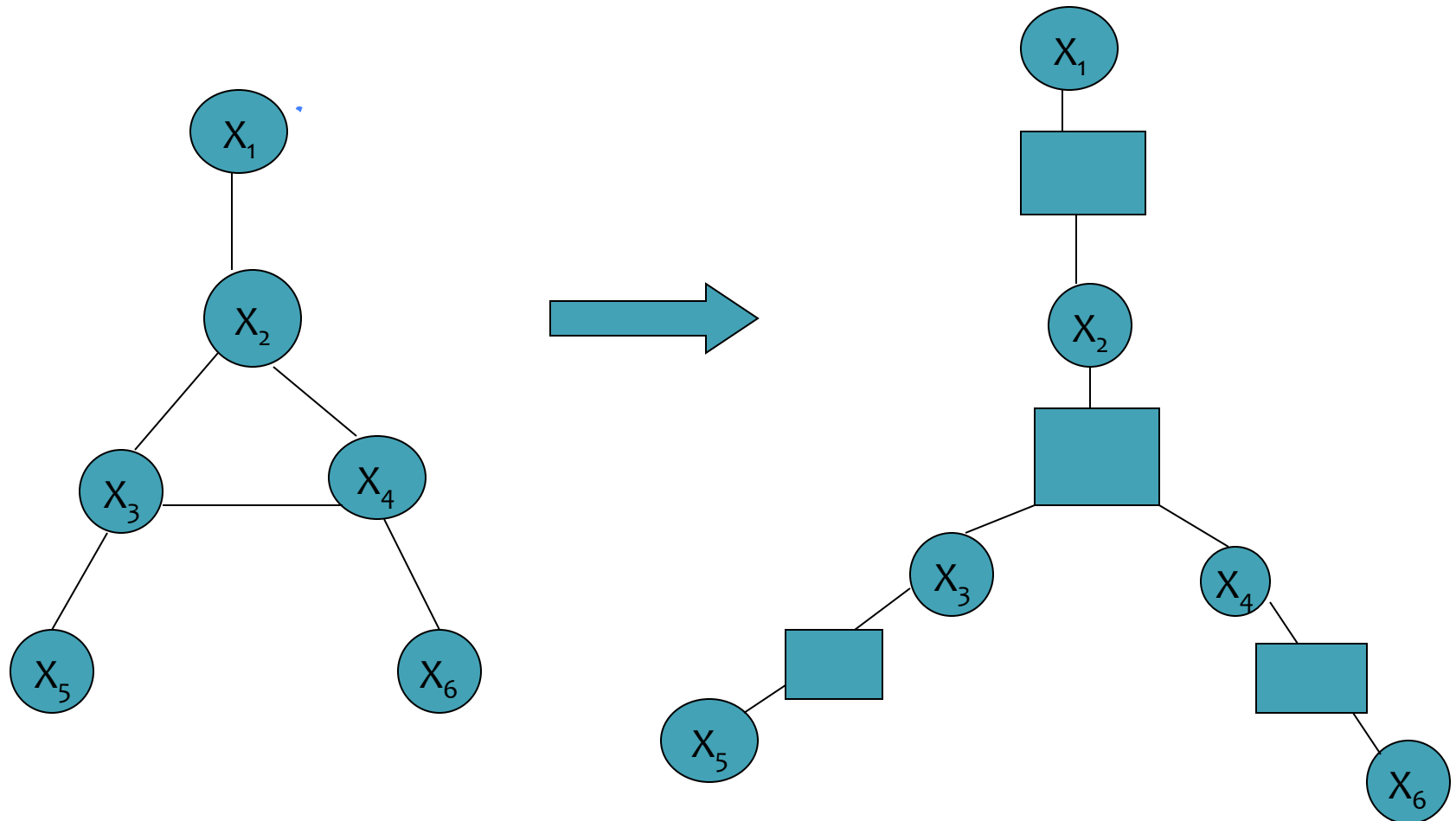
- Example 3



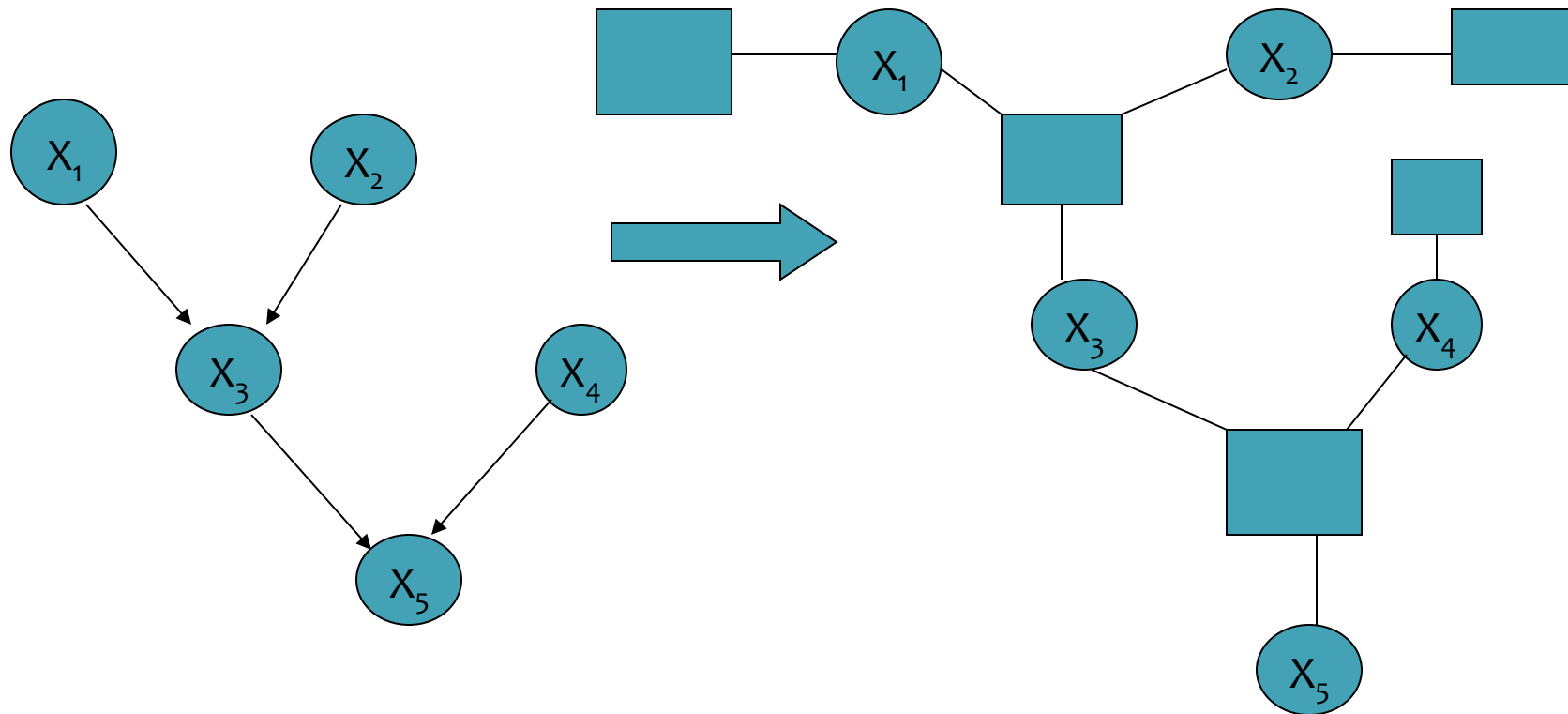
$$\psi(x_1, x_2, x_3) = f_a(x_1, x_2, x_3)$$



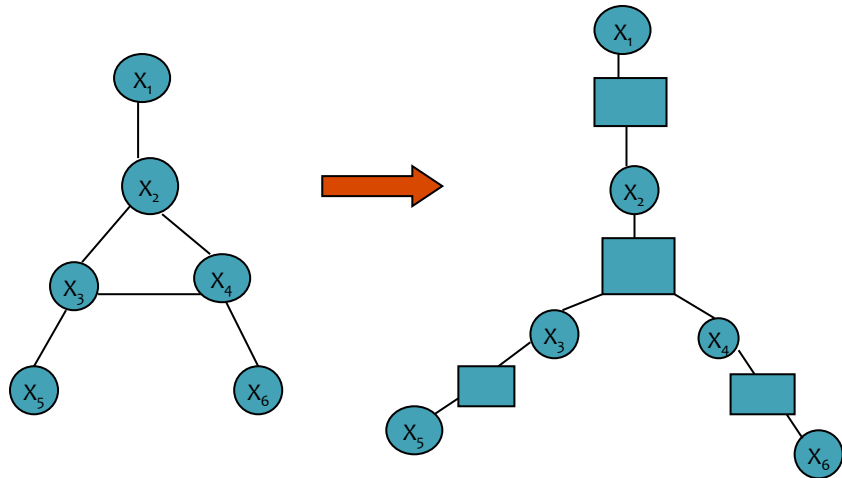
Tree-like Undirected GMs to Factor Trees



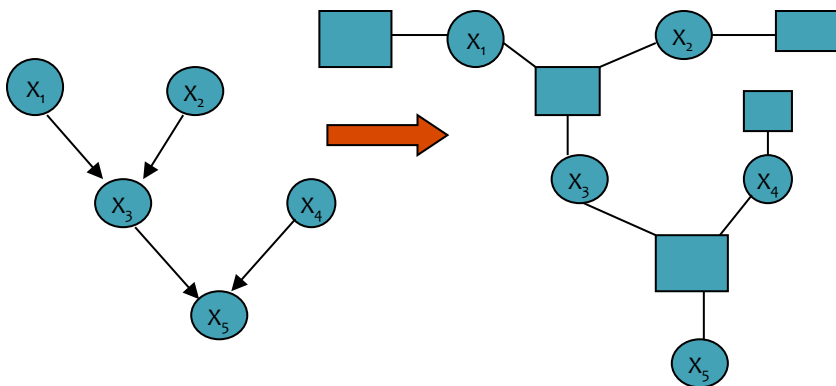
Poly-trees to Factor trees



Why factor graphs?



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



Exact Inference

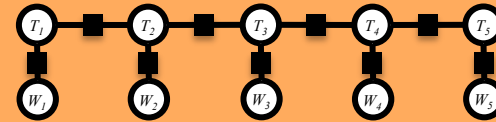
1. Data

$$\mathcal{D} = \{x^{(n)}\}_{n=1}^N$$

Sample 1:					
	time	flies	like	an	arrow
Sample 2:					
	time	flies	like	an	arrow
Sample 3:					
	flies	fly	with	their	wing
Sample 4:					
	with	time	you	will	see

2. Model

$$p(x | \theta) = \frac{1}{Z(\theta)} \prod_{C \in \mathcal{C}} \psi_C(x_C)$$



3. Objective

$$\ell(\theta; \mathcal{D}) = \sum_{n=1}^N \log p(x^{(n)} | \theta)$$

5. Inference

1. Marginal Inference

$$p(x_C) = \sum_{x': x'_C = x_C} p(x' | \theta)$$

2. Partition Function

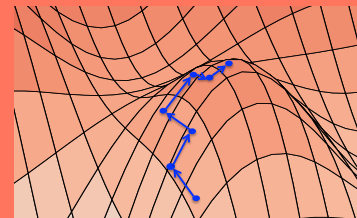
$$Z(\theta) = \sum_x \prod_{C \in \mathcal{C}} \psi_C(x_C)$$

3. MAP Inference

$$\hat{x} = \operatorname{argmax}_x p(x | \theta)$$

4. Learning

$$\theta^* = \operatorname{argmax}_{\theta} \ell(\theta; \mathcal{D})$$

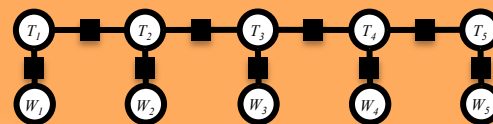


Goals for Today's Lecture

1. Unify MRFs and Bayes Nets with a new representation: **factor graphs**
2. Perform exact inference on factor graphs with two algorithms: **Elimination** and **Sum-Product Belief Propagation**

Model

$$p(\mathbf{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C)$$



Objective

$$\ell(\boldsymbol{\theta}; \mathcal{D}) = \sum_{n=1}^N \log p(\mathbf{x}^{(n)} \mid \boldsymbol{\theta})$$

5. Inference

1. Marginal Inference

$$p(\mathbf{x}_C) = \sum_{\mathbf{x}': \mathbf{x}'_C = \mathbf{x}_C} p(\mathbf{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

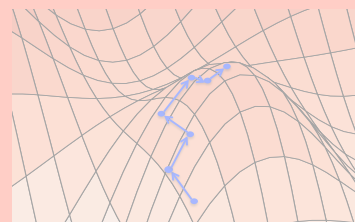
$$Z(\boldsymbol{\theta}) = \sum_{\mathbf{x}} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C)$$

3. MAP Inference

$$\hat{\mathbf{x}} = \operatorname{argmax}_{\mathbf{x}} p(\mathbf{x} \mid \boldsymbol{\theta})$$

4. Learning

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \mathcal{D})$$



5. Inference

Three Tasks: (All three are NP-Hard in the general case)

1. Marginal Inference

Compute marginals of variables and factors

$$p(x_i) = \sum_{\mathbf{x}': x'_i = x_i} p(\mathbf{x}' \mid \boldsymbol{\theta}) \quad \Bigg| \quad p(\mathbf{x}_C) = \sum_{\mathbf{x}': \mathbf{x}'_C = \mathbf{x}_C} p(\mathbf{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

Compute the normalization constant

$$Z(\boldsymbol{\theta}) = \sum_{\mathbf{x}} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C)$$

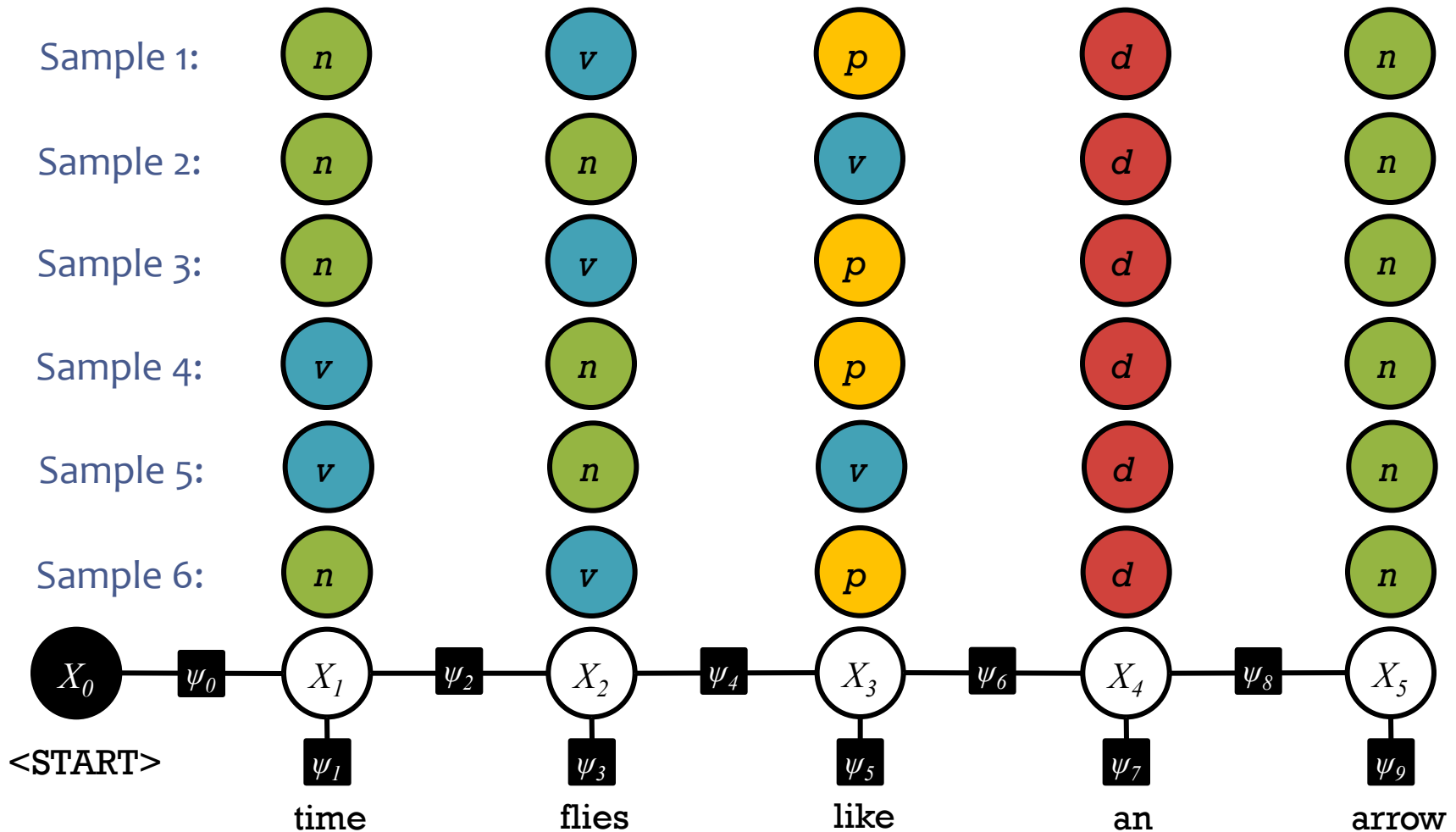
3. MAP Inference

Compute variable assignment with highest probability

$$\hat{\mathbf{x}} = \operatorname{argmax}_{\mathbf{x}} p(\mathbf{x} \mid \boldsymbol{\theta})$$

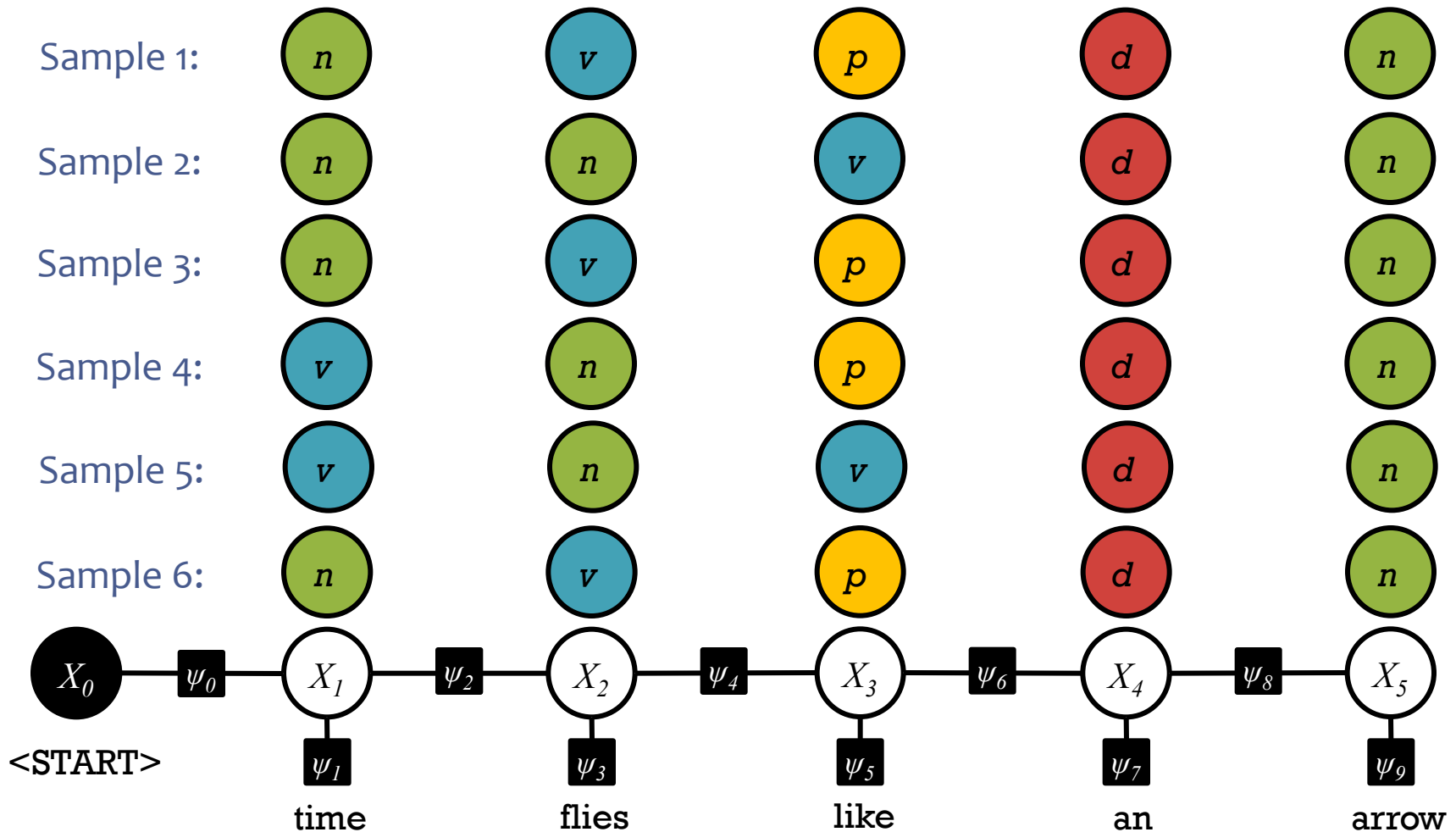
Marginals by Sampling on Factor Graph

Suppose we took many samples from the distribution over taggings: $p(\mathbf{x}) = \frac{1}{Z} \prod_{\alpha} \psi_{\alpha}(\mathbf{x}_{\alpha})$



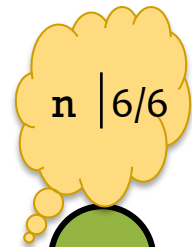
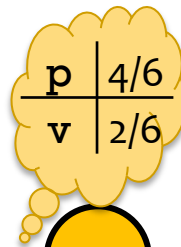
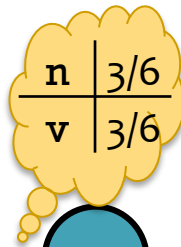
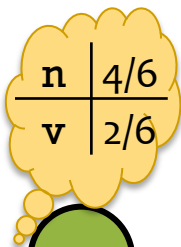
Marginals by Sampling on Factor Graph

The marginal $p(X_i = x_i)$ gives the probability that variable X_i takes value x_i in a random sample



Marginals by Sampling on Factor Graph

Estimate the
marginals as:



Sample 1:

n

v

p

d

n

Sample 2:

n

n

v

d

n

Sample 3:

n

v

p

d

n

Sample 4:

v

n

p

d

n

Sample 5:

v

n

v

d

n

Sample 6:

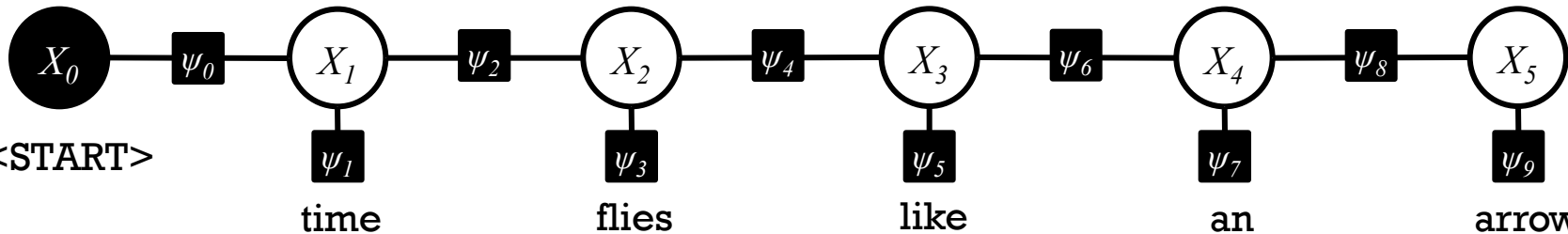
n

v

p

d

n



How do we get marginals without sampling?

That's what Belief Propagation is all about!

Why not just sample?

- Sampling one joint assignment is *also* NP-hard in general.
 - In practice: Use MCMC (e.g., Gibbs sampling) as an anytime algorithm.
 - So draw an approximate sample fast, or run longer for a “good” sample.
- Sampling finds the high-probability values x_i efficiently.
But it takes too many samples to see the low-probability ones.
 - How do you find $p(\text{“The quick brown fox ...”})$ under a language model?
 - Draw random sentences to see how often you get it? Takes a *long* time.
 - Or multiply factors (trigram probabilities)? That's what BP would do.

Simple and general exact inference for graphical models

VARIABLE ELIMINATION

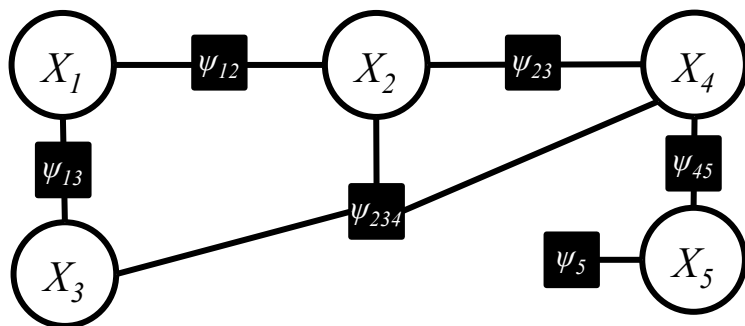
The Variable Elimination Algorithm

For all i , suppose the **range** of X_i is $\{0, 1, 2\}$.

Let $k=3$ denote the **size of the range**.

The distribution **factorizes** as:

$$p(\mathbf{x}) = \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{24}(x_2, x_4) \\ \psi_{234}(x_2, x_3, x_4) \psi_{45}(x_4, x_5) \psi_5(x_5)$$



Naively, we compute the **partition function**

as:

$$Z = \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} p(\mathbf{x})$$

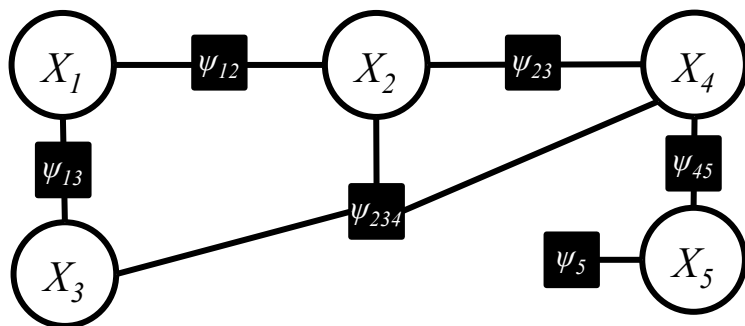
The Variable Elimination Algorithm

For all i , suppose the **range** of X_i is $\{0, 1, 2\}$.

Let $k=3$ denote the **size of the range**.

The distribution **factorizes** as:

$$p(\mathbf{x}) = \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{24}(x_2, x_4) \\ \psi_{234}(x_2, x_3, x_4) \psi_{45}(x_4, x_5) \psi_5(x_5)$$



Naively, we compute the **partition function** as:

$$Z = \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} p(\mathbf{x})$$

$p(\mathbf{x})$ can be represented as a joint probability table with 3^5 entries:

x_1	x_2	x_3	x_4	x_5	$p(\mathbf{x})$
0	0	0	0	0	0.019517693
0	0	0	0	1	0.017090249
0	0	0	0	2	0.014885825
0	0	0	1	0	0.024117638
0	0	0	1	1	0.000925849
0	0	0	1	2	0.028112576
0	0	0	2	0	0.028050205
0	0	0	2	1	0.004812689
0	0	0	2	2	0.007987737
0	0	1	0	0	0.028433687
0	0	1	0	1	0.037073469
0	0	1	0	2	0.013558227
0	0	1	1	0	0.019479016
0	0	1	1	1	0.012312901
0	0	1	1	2	0.023439775
0	0	1	2	0	0.038206131
0	0	1	2	1	0.038996005
0	0	1	2	2	0.041458783
0	0	2	0	0	0.044616806
0	0	2	0	1	0.020846989
0	0	2	0	2	0.03006475
0	0	2	1	0	0.048436964
0	0	2	1	1	0.02854376
0	0	2	1	2	0.029191506
0	0	2	2	0	0.031531118
0	0	2	2	1	0.005132392
0	0	2	2	2	0.032027091
...

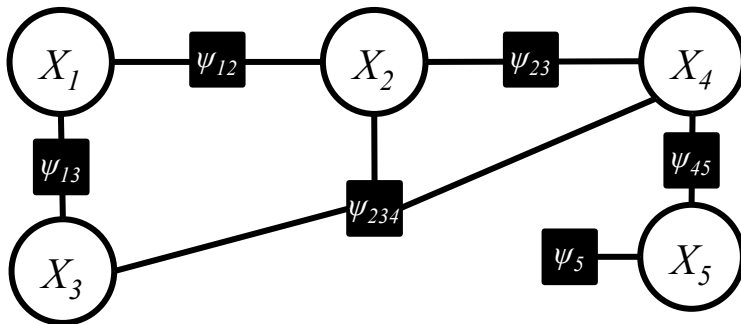
The Variable Elimination Algorithm

For all i , suppose the **range** of X_i is $\{0, 1, 2\}$.

Let $k=3$ denote the **size of the range**.

The distribution **factorizes** as:

$$p(\mathbf{x}) = \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{24}(x_2, x_4) \\ \psi_{234}(x_2, x_3, x_4) \psi_{45}(x_4, x_5) \psi_5(x_5)$$



Naively, we compute the **partition function** as:

$$Z = \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} p(\mathbf{x})$$

$p(\mathbf{x})$ can be represented as a joint probability table with 3^5 entries:

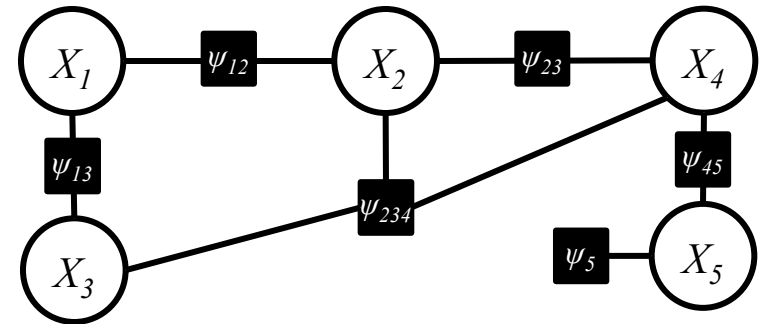
x_1	x_2	x_3	x_4	x_5	$p(\mathbf{x})$
0	0	0	0	0	0.019517693
0	0	0	0	1	0.017090249
0	0	0	0	2	0.014885825
0	0	0	1	0	0.024117638
0	0	0	1	1	0.000925849
0	0	0	1	2	0.028112576
0	0	0	2	0	0.028050205
0	0	0	2	1	0.004812689
0	0	0	2	2	0.007987737
0	0	1	0	0	0.028433687
0	0	1	0	1	0.037073469
0	0	1	0	2	0.013558227
0	0	1	1	0	0.019479016
0	0	1	1	1	0.012312901
0	0	1	1	2	0.023439775
0	0	1	2	0	0.038206131
0	0	1	2	1	0.038996005
0	0	1	2	2	0.041458783
0	0	2	0	0	0.044616806
0	0	2	0	1	0.020846989
0	0	2	0	2	0.03006475
0	0	2	1	0	0.048436964
0	0	2	1	1	0.02854376

Naïve computation of Z requires 3^5 additions.

Can we do better?

The Variable Elimination Algorithm

Instead, capitalize on the factorization of $p(\mathbf{x})$.



$$\begin{aligned}
 Z &= \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) \psi_{45}(x_4, x_5) \psi_5(x_5) \\
 &= \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) \underbrace{\sum_{x_5} \psi_{45}(x_4, x_5) \psi_5(x_5)}_{\text{factor}}
 \end{aligned}$$

Only 3^2 additions are needed to marginalize out x_5 .

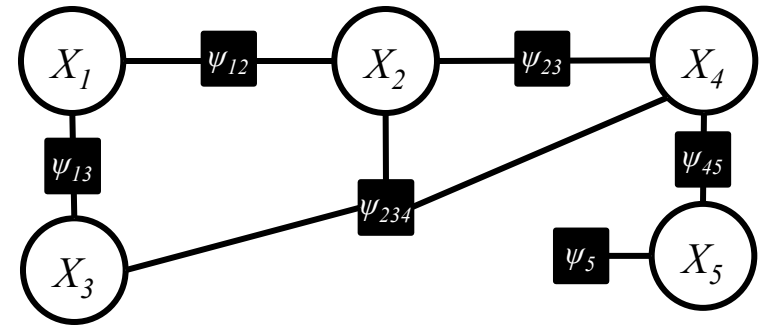
We denote the marginal's table by $m_5(x_4)$.

This “factor” is a much smaller table with 3^2 entries:

x_4	x_5	$p(\mathbf{x})$
0	0	0.019517693
0	1	0.017090249
0	2	0.014885825
1	0	0.024117638
1	1	0.000925849
1	2	0.028112576
2	0	0.028050205
2	1	0.004812689
2	2	0.007987737

The Variable Elimination Algorithm

Instead, capitalize on the factorization of $p(\mathbf{x})$.

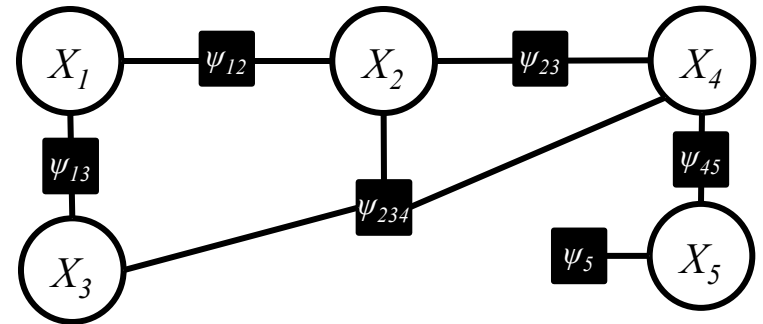


$$\begin{aligned}
 Z &= \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) \psi_{45}(x_4, x_5) \psi_5(x_5) \\
 &= \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) \sum_{x_5} \psi_{45}(x_4, x_5) \psi_5(x_5) \\
 &= \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) m_5(x_4)
 \end{aligned}$$

$$m_5(x_4) \triangleq \sum_{x_5} \psi_{45}(x_4, x_5) \psi_5(x_5)$$

The Variable Elimination Algorithm

Instead, capitalize on the factorization of $p(\mathbf{x})$.



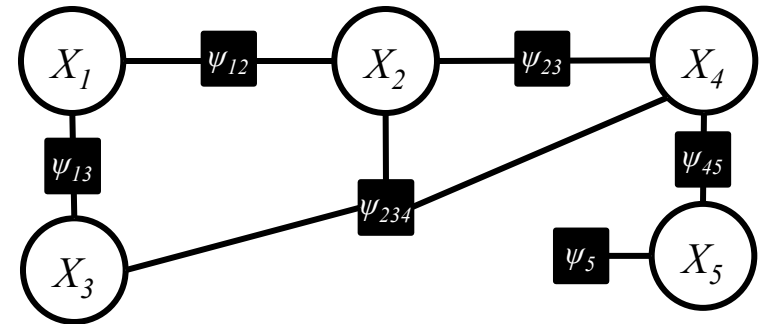
$$\begin{aligned}
 Z &= \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) \psi_{45}(x_4, x_5) \psi_5(x_5) \\
 &= \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) \sum_{x_5} \psi_{45}(x_4, x_5) \psi_5(x_5) \\
 &= \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \underbrace{\psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4)}_{m_5(x_4)} m_5(x_4)
 \end{aligned}$$

This “factor” is still a 3^4 table so apply the same trick again.

$$m_5(x_4) \triangleq \sum_{x_5} \psi_{45}(x_4, x_5) \psi_5(x_5)$$

The Variable Elimination Algorithm

Instead, capitalize on the factorization of $p(\mathbf{x})$.



$$\begin{aligned}
 Z &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) \sum_{x_5} \psi_{45}(x_4, x_5) \psi_5(x_5) \\
 &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) m_5(x_4) \\
 &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) m_4(x_2, x_3) \\
 &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) m_3(x_1, x_2) \\
 &= \sum_{x_1} m_2(x_1)
 \end{aligned}$$

3^2 additions

3^3 additions

3^3 additions

3^2 additions

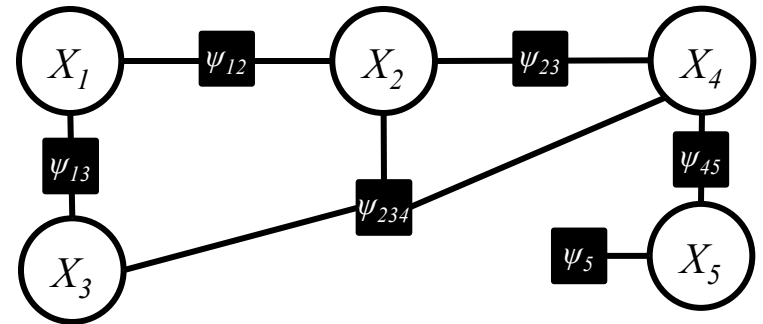
3 additions

Naïve solution requires $3^5 = 243$ additions.

Variable elimination only requires $3 + 3^2 + 3^3 + 3^3 + 3^2 = 75$ additions.

The Variable Elimination Algorithm

The same trick can be used to compute **marginal probabilities**. Just choose the variable elimination order such that the query variables are last.



$$\begin{aligned}
 p(x_1) &= \frac{1}{Z} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) \sum_{x_5} \psi_{45}(x_4, x_5) \psi_5(x_5) \\
 &= \frac{1}{Z} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) m_5(x_4) \\
 &= \frac{1}{Z} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) m_4(x_2, x_3) \\
 &= \frac{1}{Z} \sum_{x_2} \psi_{12}(x_1, x_2) m_3(x_1, x_2) \\
 &= \frac{1}{Z} m_2(x_1)
 \end{aligned}$$

3^2 additions

3^3 additions

3^3 additions

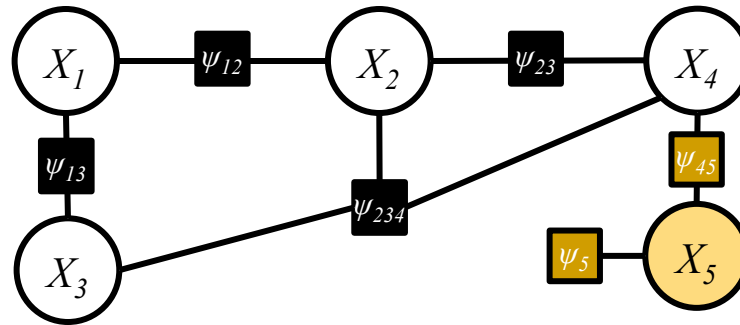
3^2 additions

3 different values on LHS

For directed graphs, $Z = 1$.

For undirected graphs, if we compute each (unnormalized) value on the LHS, we can sum them to get Z .

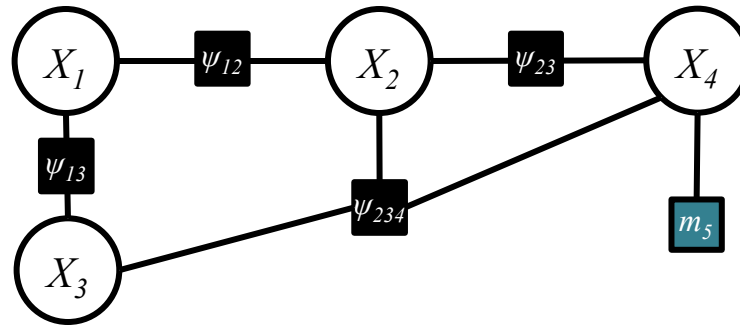
The Variable Elimination Algorithm



$$\begin{aligned}
 Z &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) \sum_{x_5} \psi_{45}(x_4, x_5) \psi_5(x_5) \\
 &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) m_5(x_4) \\
 &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) m_4(x_2, x_3)
 \end{aligned}$$

In a factor graph, variable **elimination** corresponds to replacement of a subgraph with a factor.

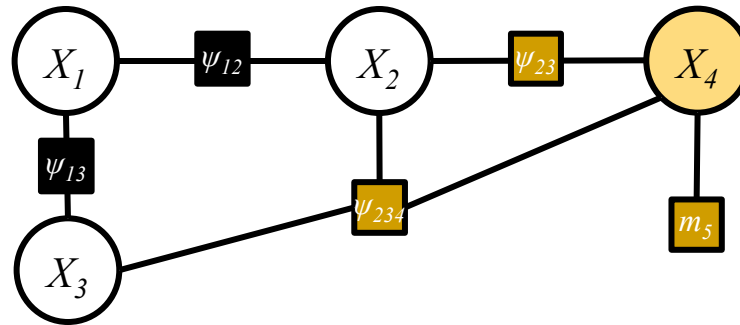
The Variable Elimination Algorithm



$$\begin{aligned}
 Z &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) \sum_{x_5} \psi_{45}(x_4, x_5) \psi_5(x_5) \\
 &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) m_5(x_4) \\
 &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) m_4(x_2, x_3)
 \end{aligned}$$

In a factor graph, variable **elimination** corresponds to replacement of a subgraph with a factor.

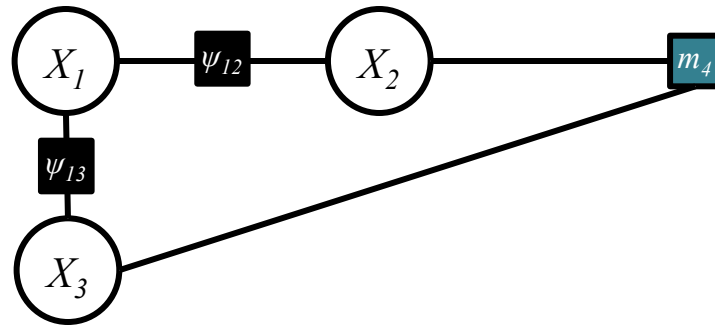
The Variable Elimination Algorithm



$$\begin{aligned}
 Z &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) \sum_{x_5} \psi_{45}(x_4, x_5) \psi_5(x_5) \\
 &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) m_5(x_4) \\
 &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) m_4(x_2, x_3)
 \end{aligned}$$

In a factor graph, variable **elimination** corresponds to replacement of a subgraph with a factor.

The Variable Elimination Algorithm

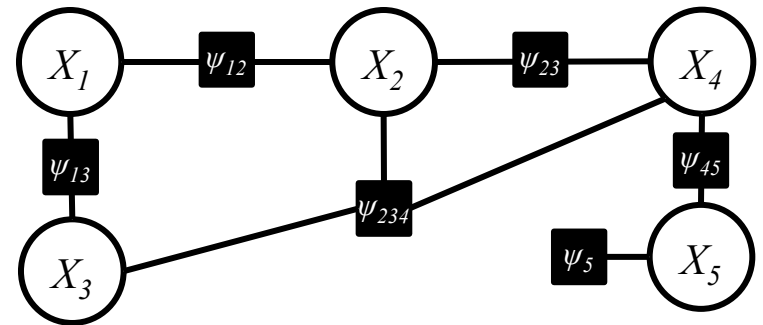


$$\begin{aligned}
 Z &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) \sum_{x_5} \psi_{45}(x_4, x_5) \psi_5(x_5) \\
 &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) m_5(x_4) \\
 &= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) m_4(x_2, x_3)
 \end{aligned}$$

In a factor graph, variable **elimination** corresponds to replacement of a subgraph with a factor.

Variable Elimination Complexity

Instead, capitalize on the factorization of $p(\mathbf{x})$.



Naïve solution is
 $O(k^n)$

Variable elimination
is $O(nk^r)$

where

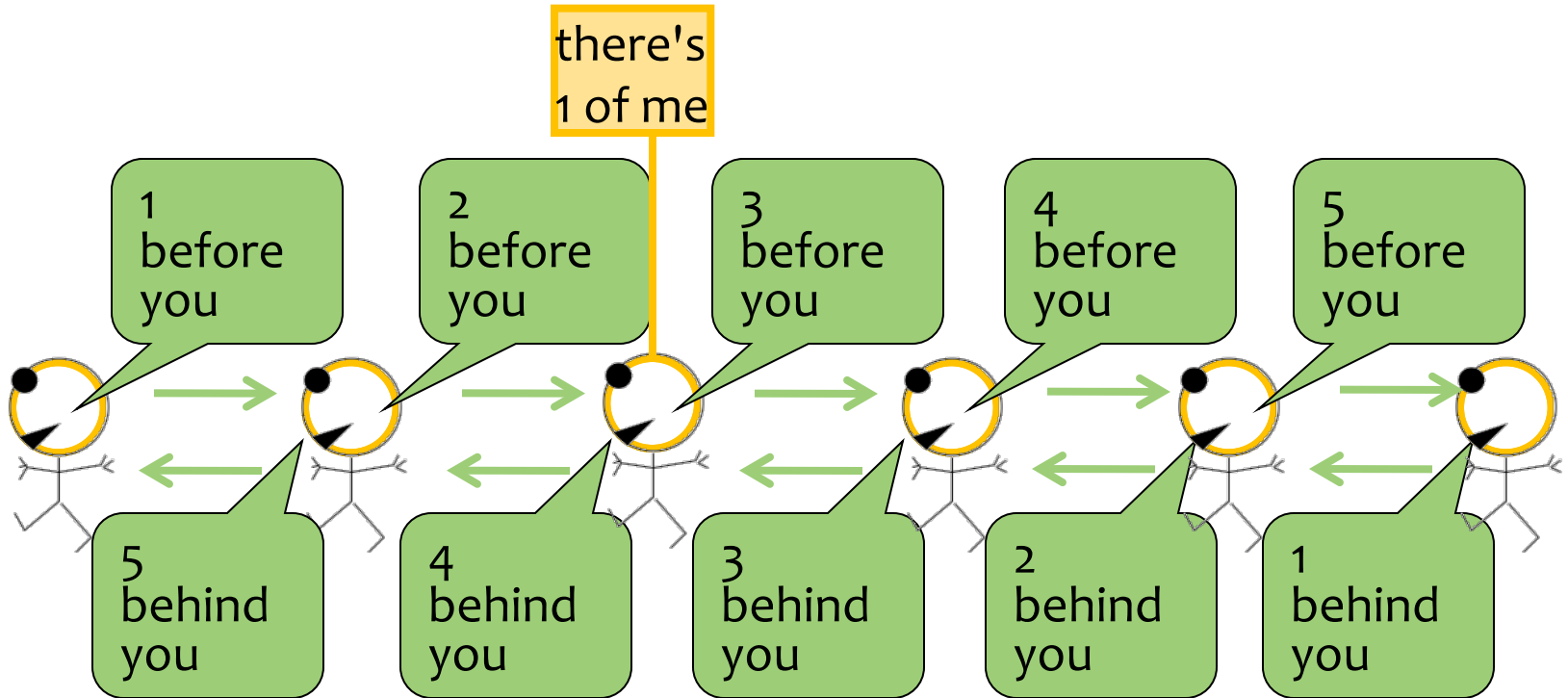
- n = # of variables
- k = max # values a variable can take
- r = # variables participating in largest “intermediate” table

Exact marginal inference for factor trees

SUM-PRODUCT BELIEF PROPAGATION

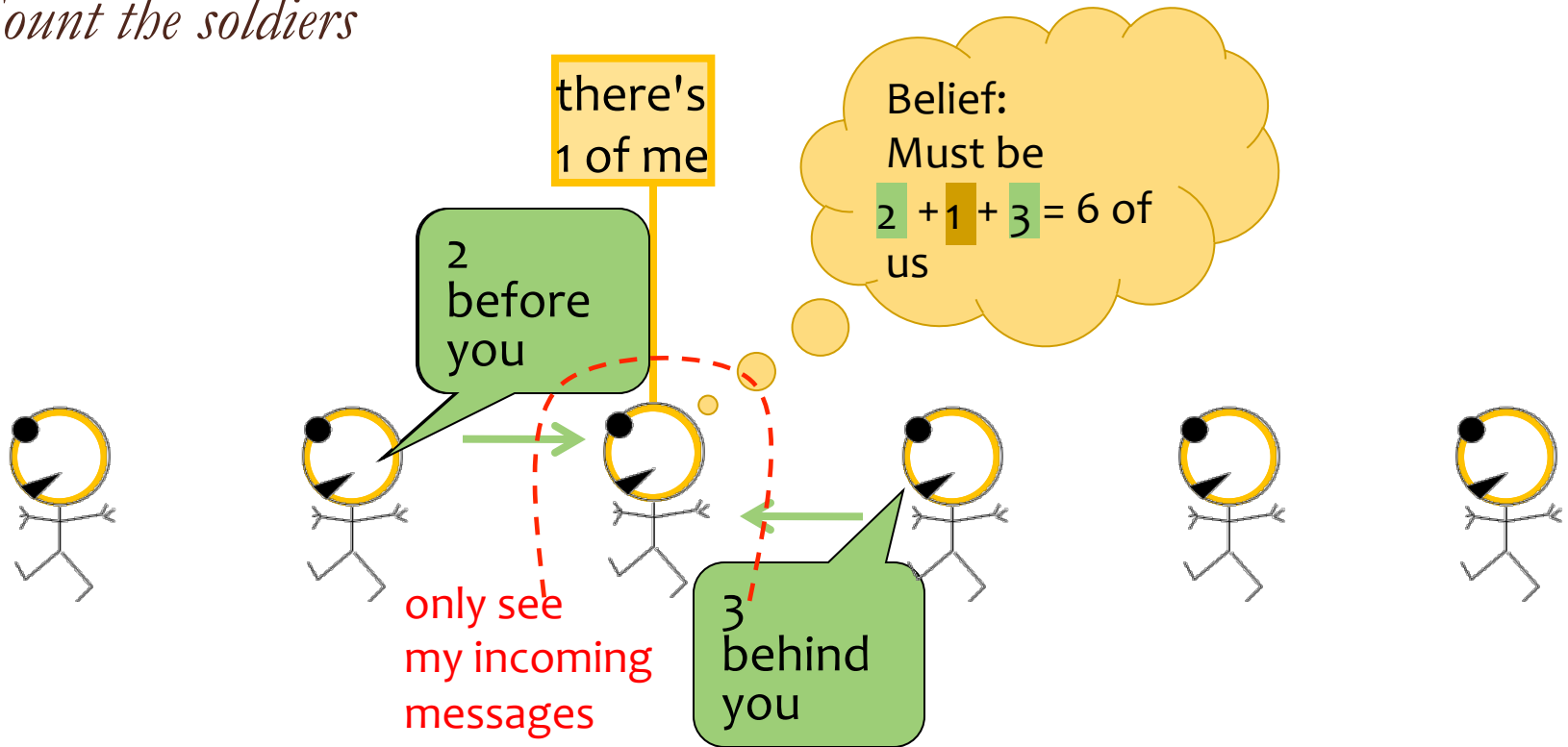
Great Ideas in ML: Message Passing

Count the soldiers



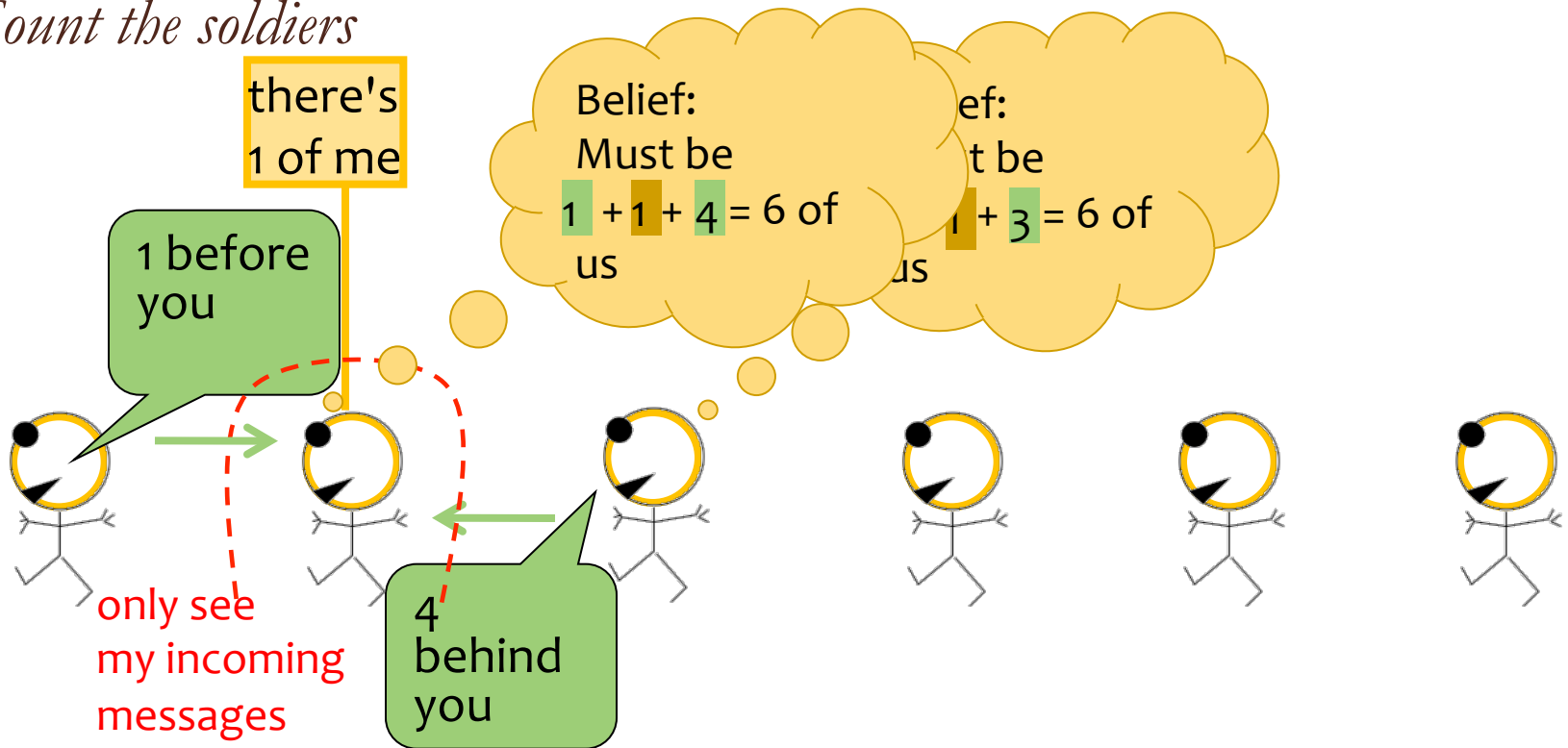
Great Ideas in ML: Message Passing

Count the soldiers



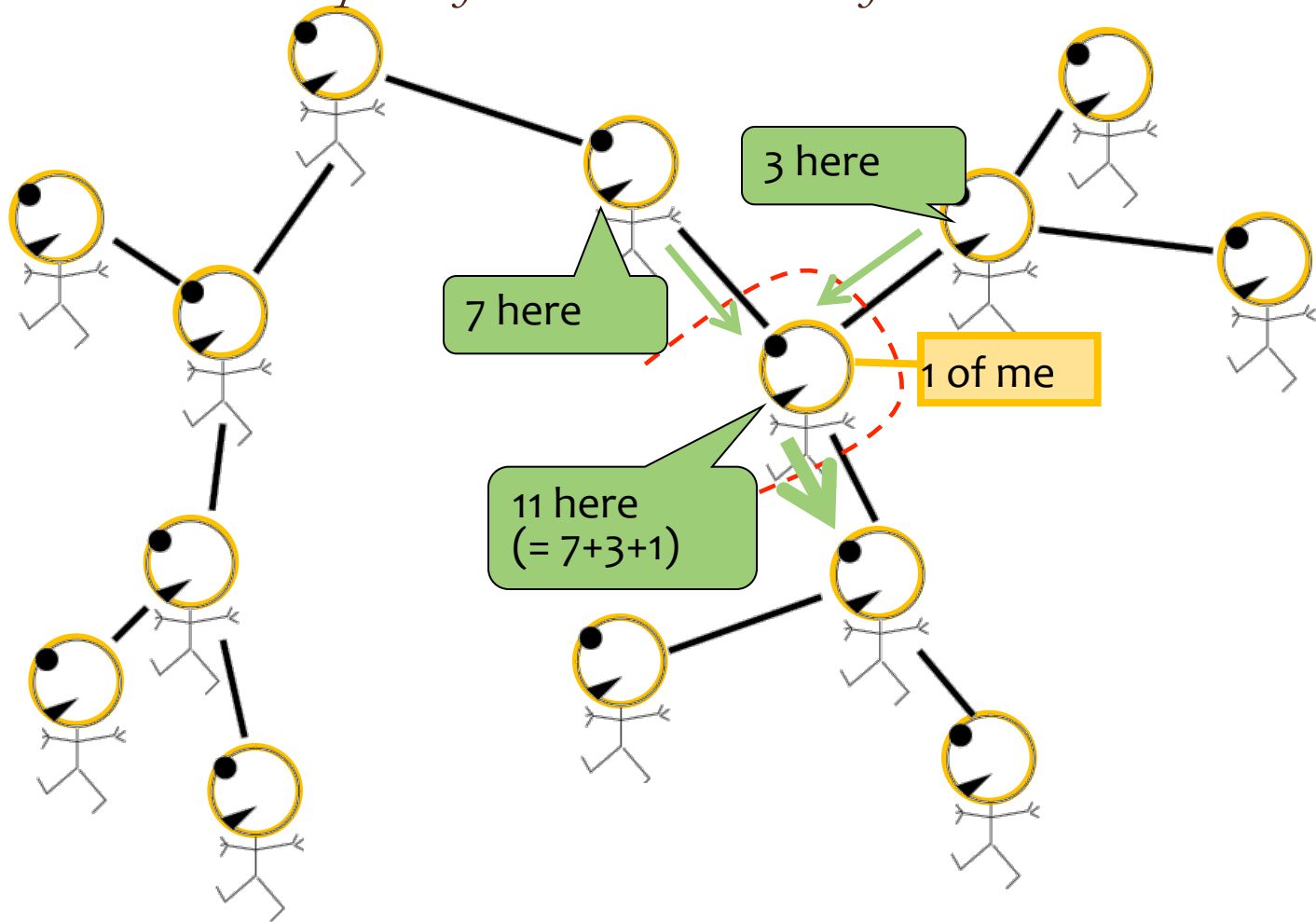
Great Ideas in ML: Message Passing

Count the soldiers



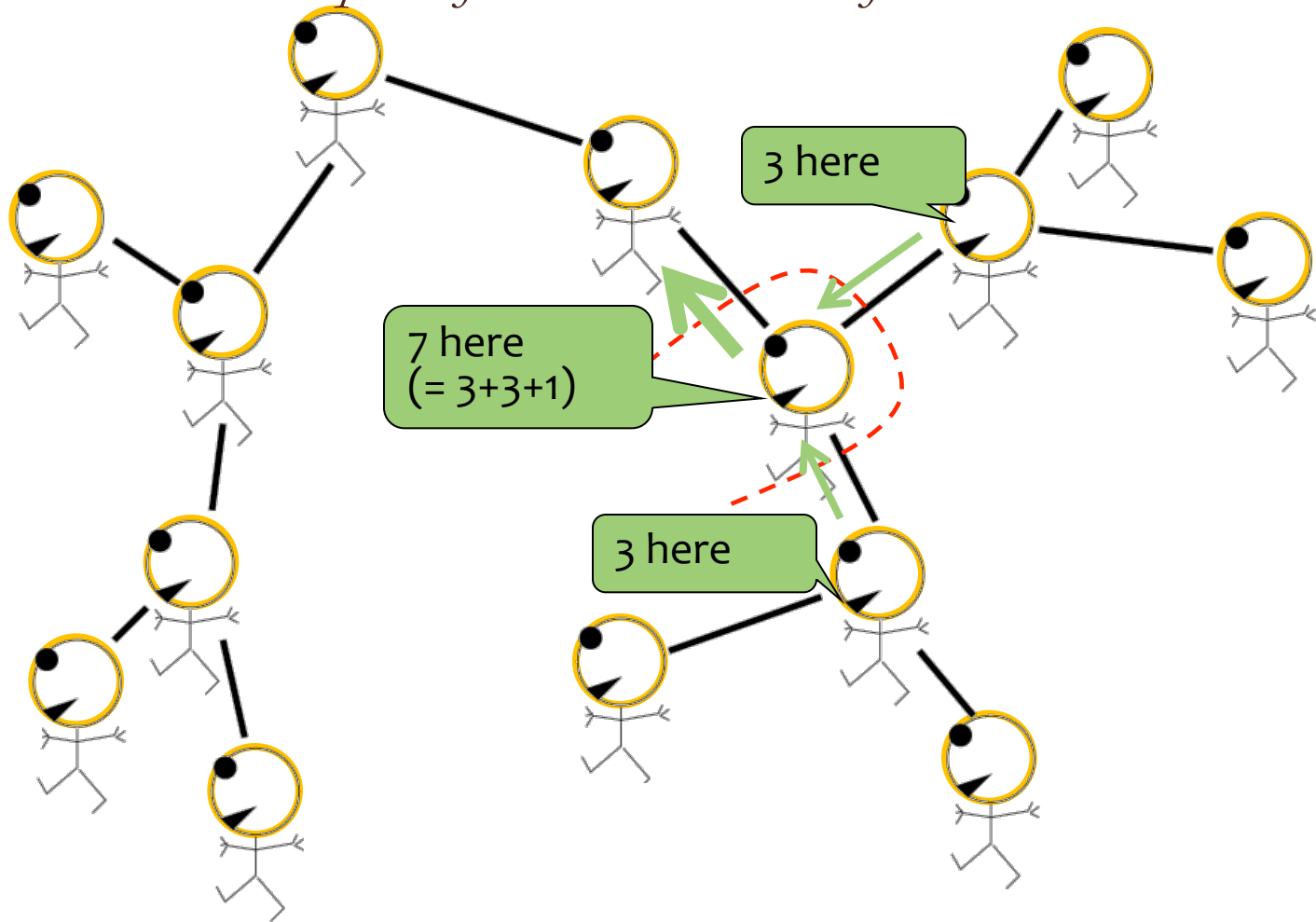
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree



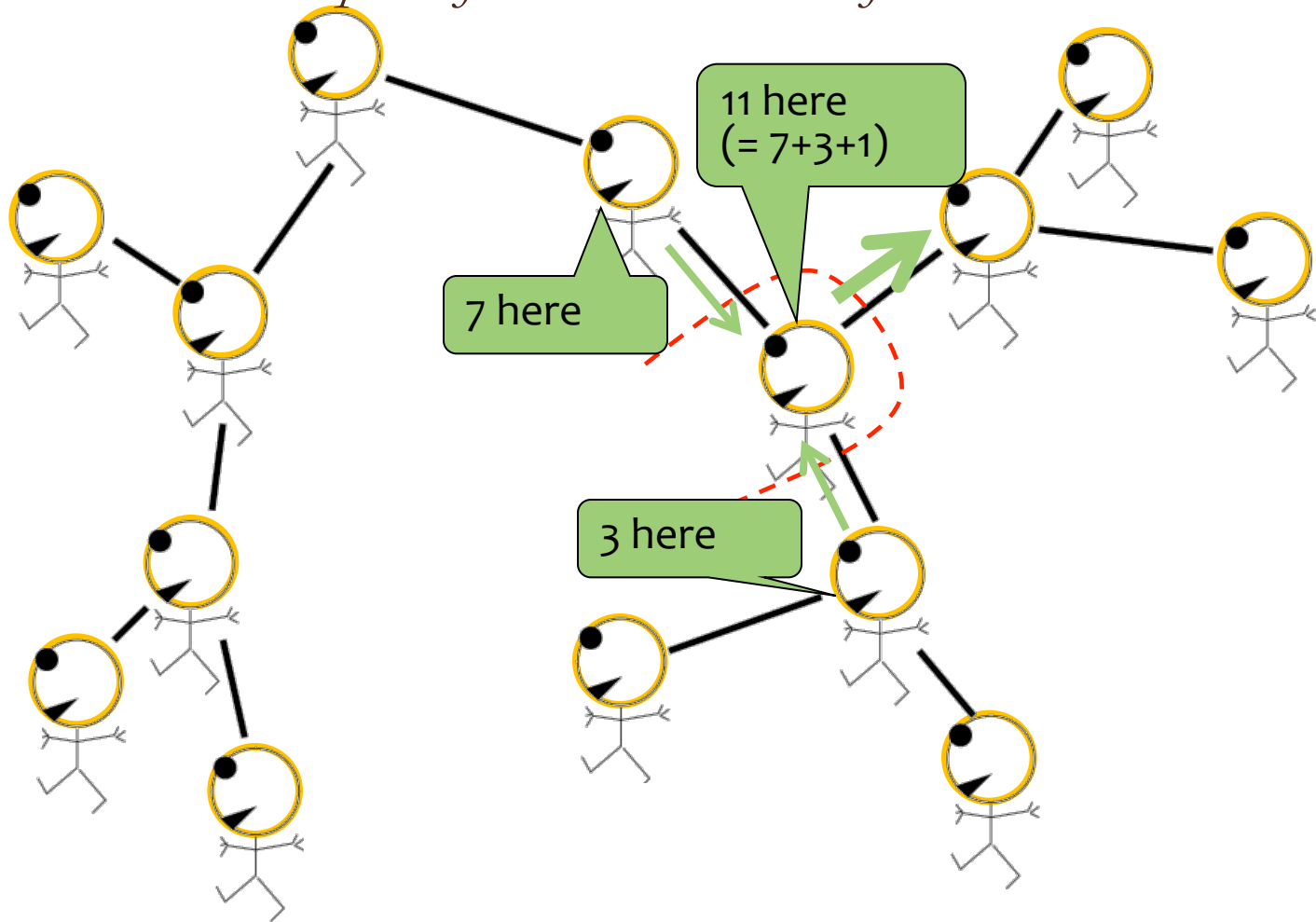
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree



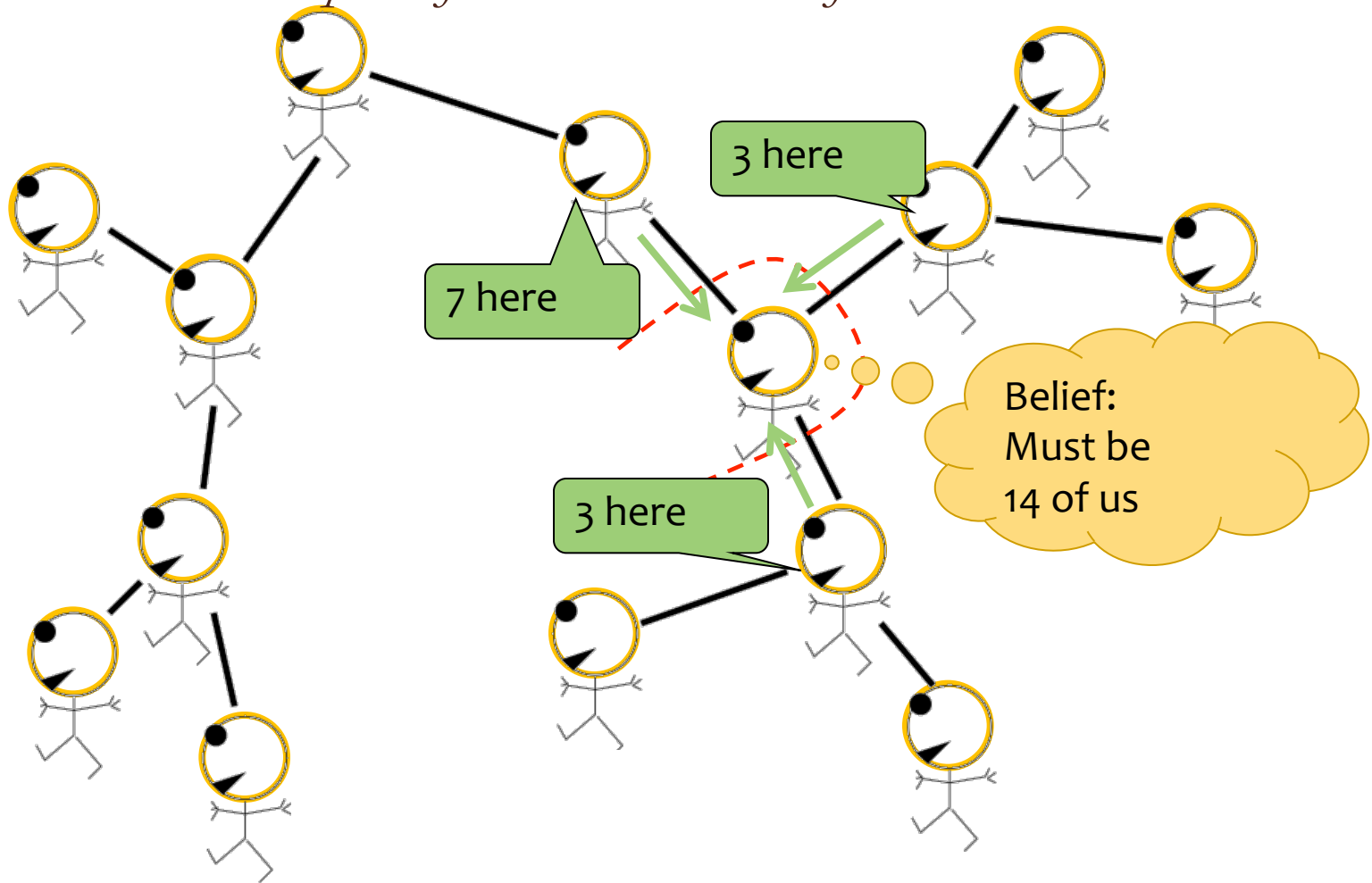
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree



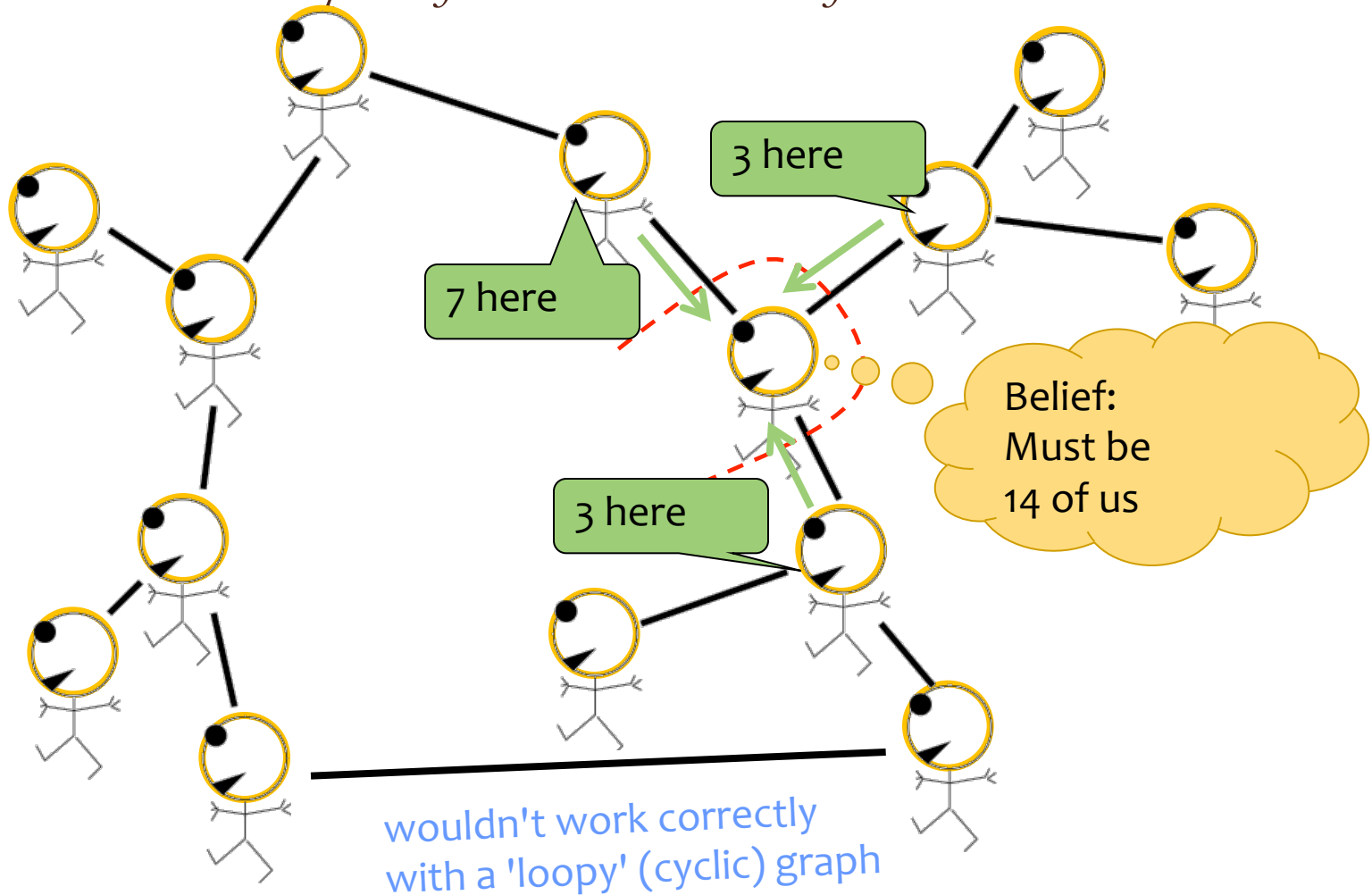
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

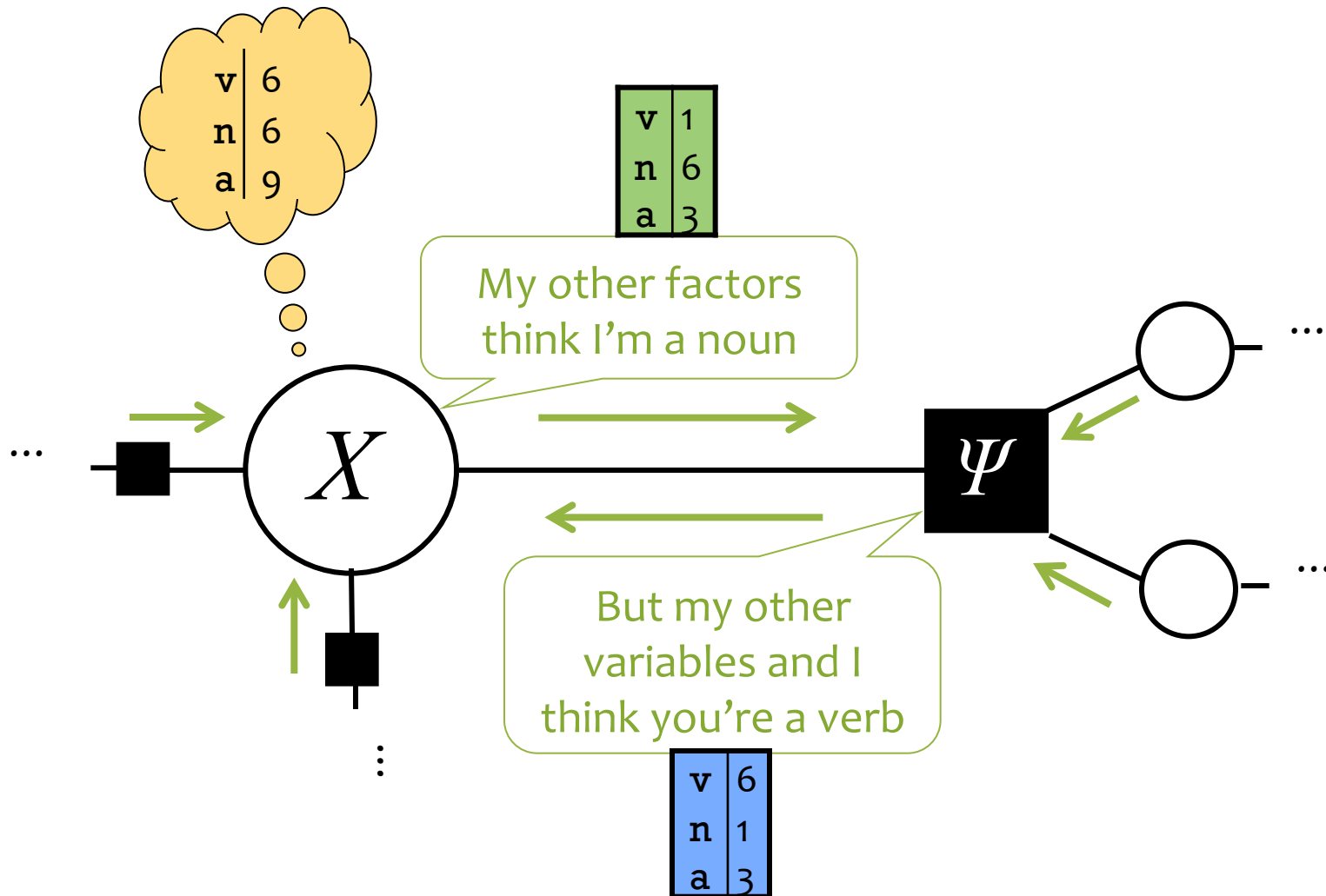


Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

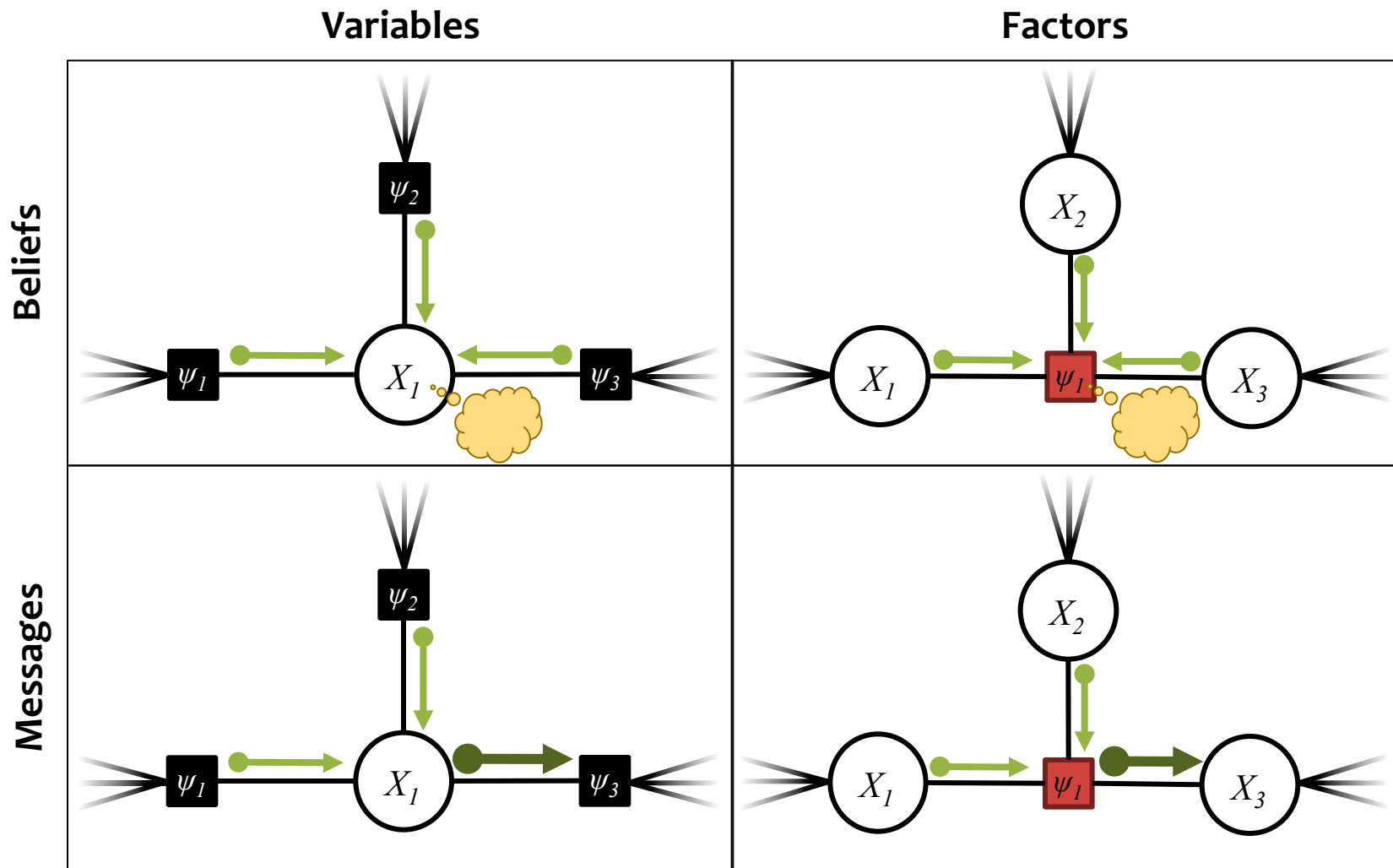


Message Passing in Belief Propagation



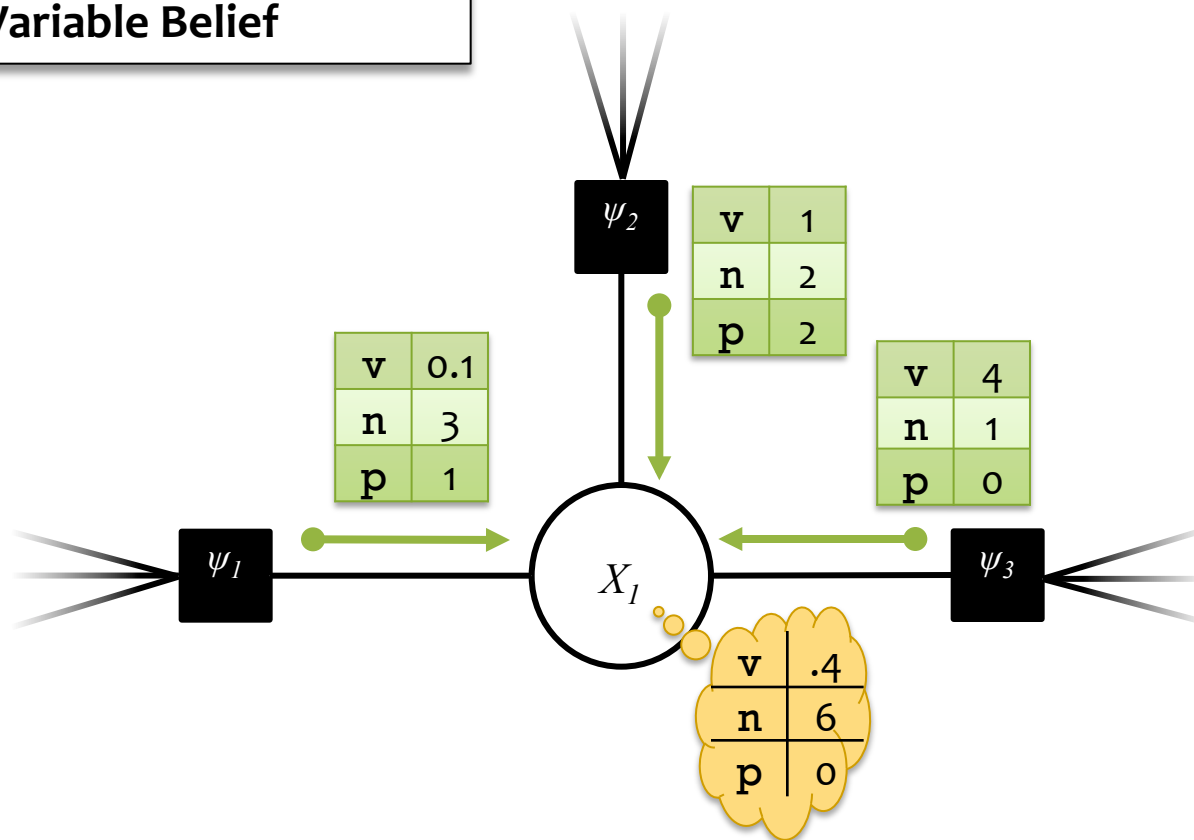
Both of these messages judge the possible values of variable X .
Their product = belief at X = product of all 3 messages to X .

Sum-Product Belief Propagation



Sum-Product Belief Propagation

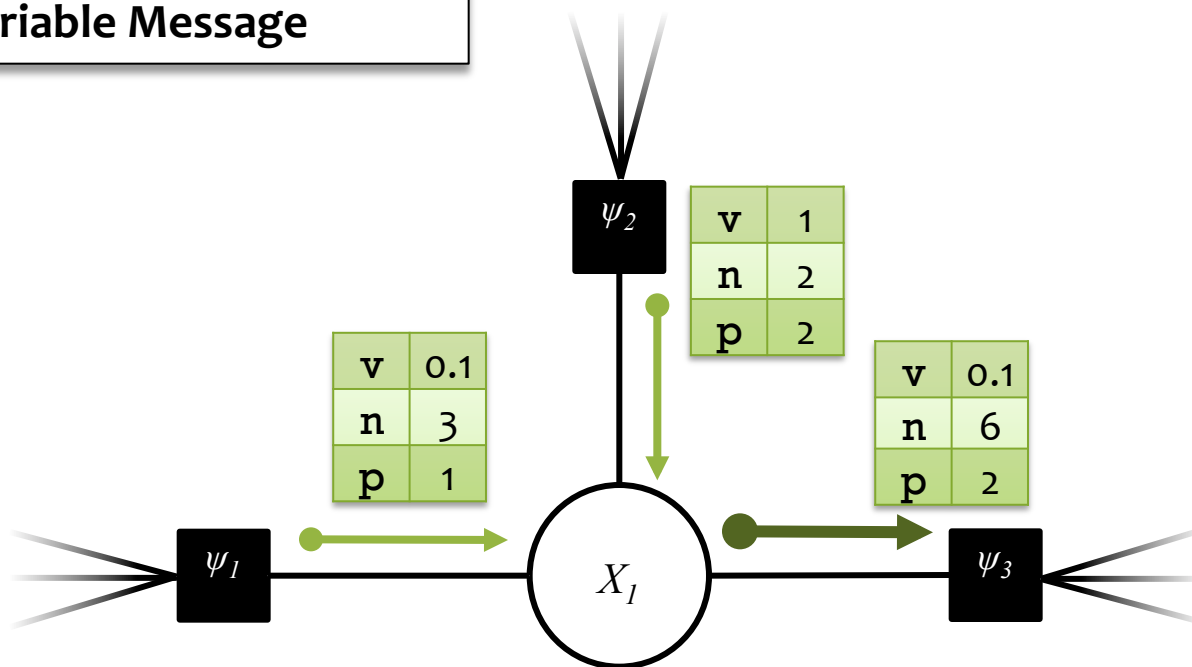
Variable Belief



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

Sum-Product Belief Propagation

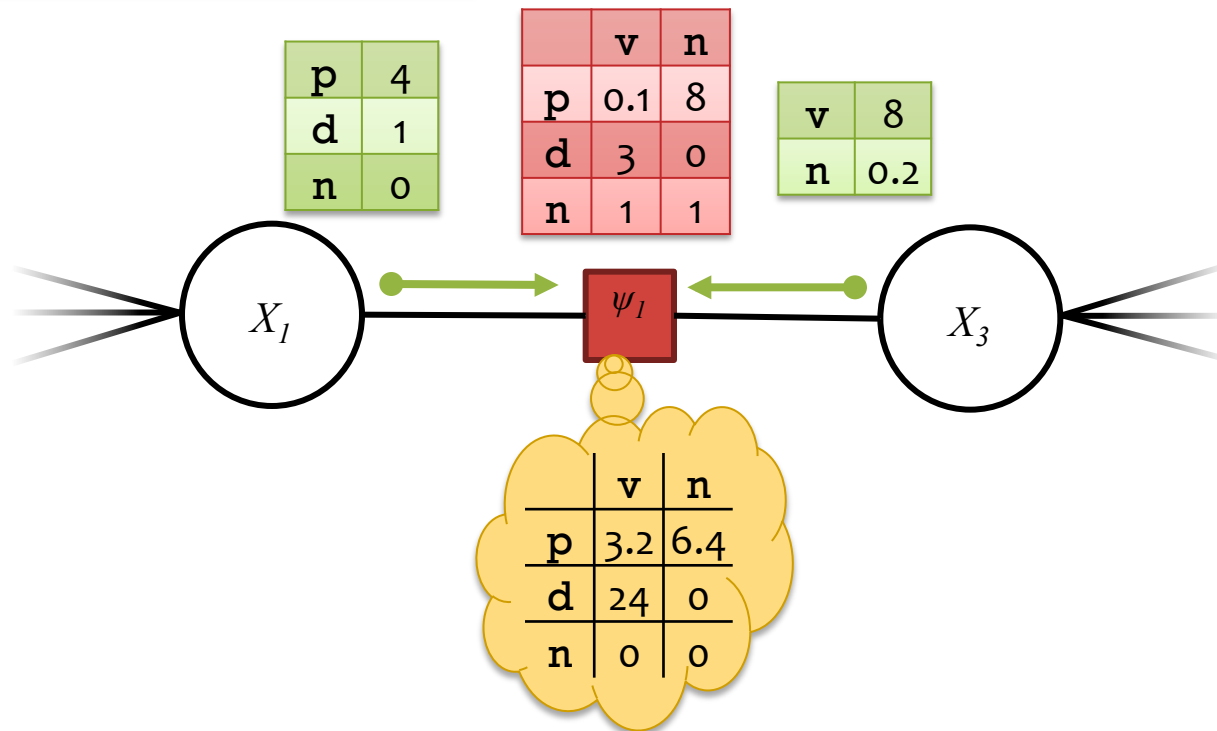
Variable Message



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

Sum-Product Belief Propagation

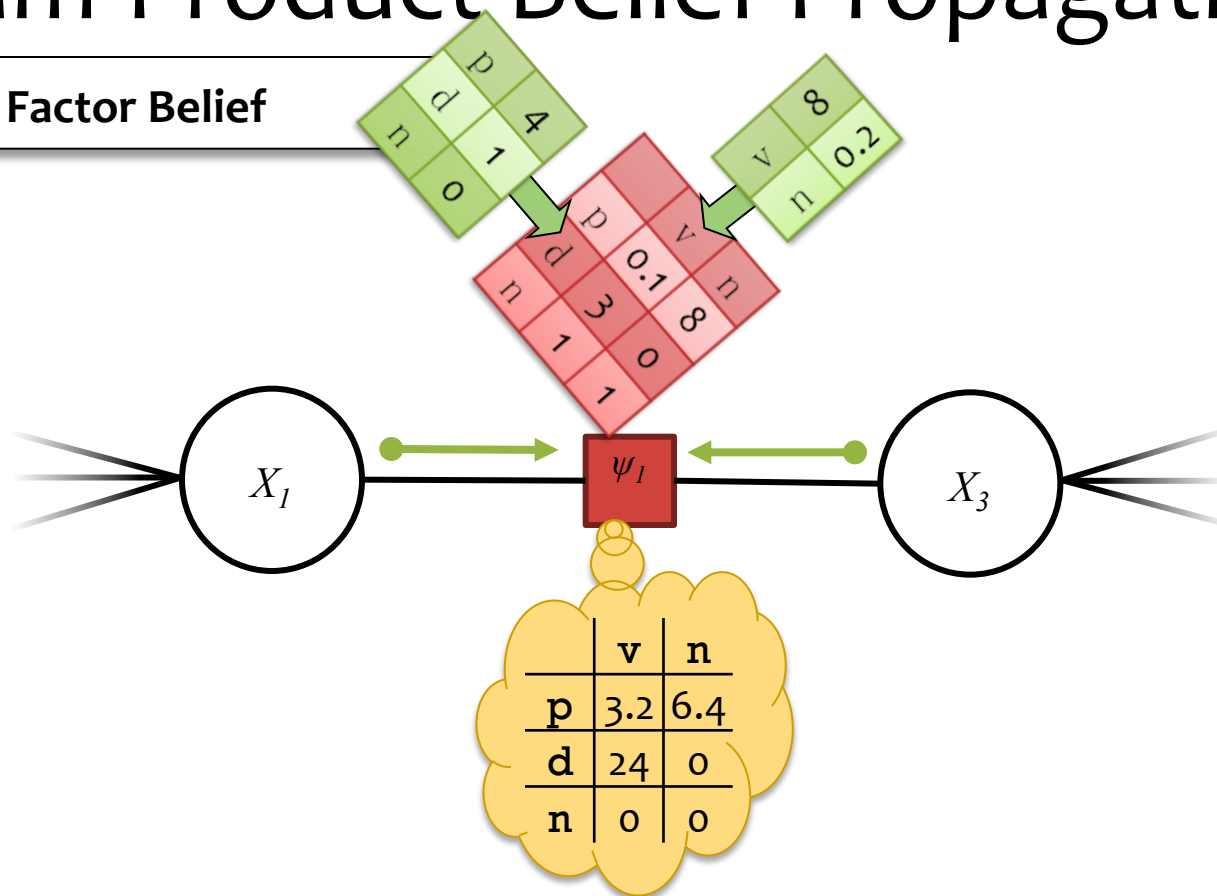
Factor Belief



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

Sum-Product Belief Propagation

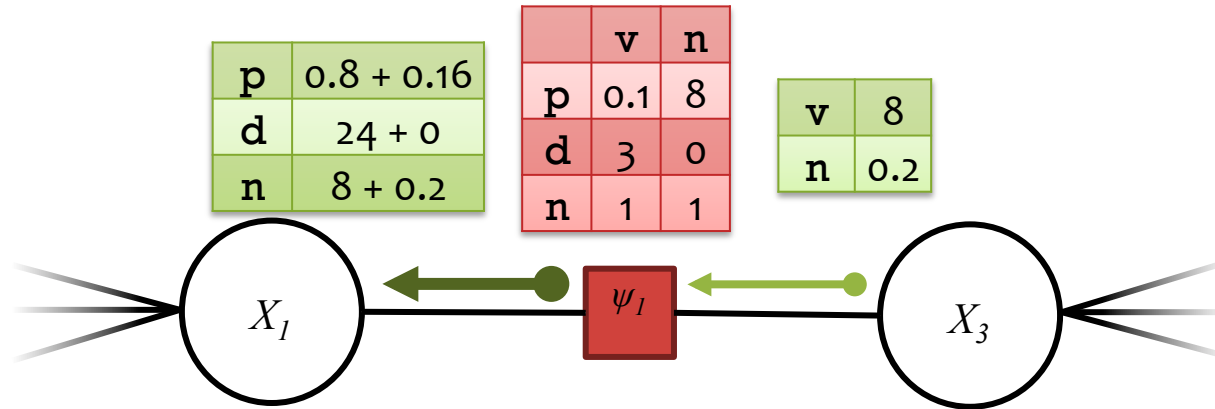
Factor Belief



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

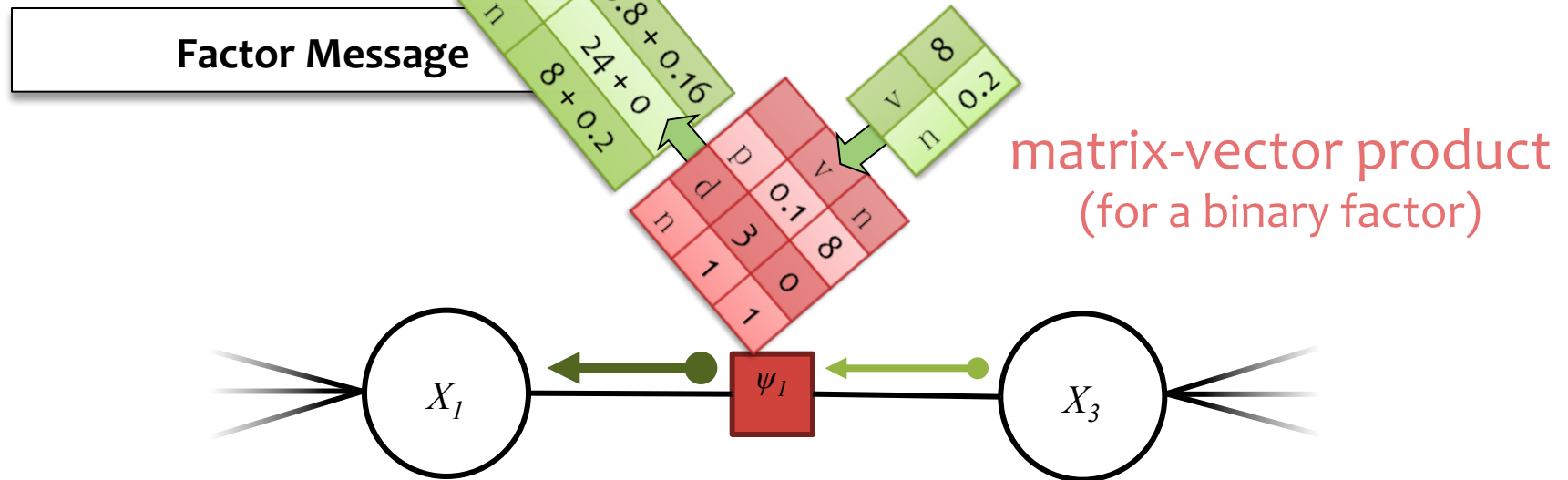
Sum-Product Belief Propagation

Factor Message



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

Sum-Product Belief Propagation



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

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.

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

2. Choose a root node.

3. Send messages from the **leaves** to the **root**.
Send messages from the **root** to the **leaves**.

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

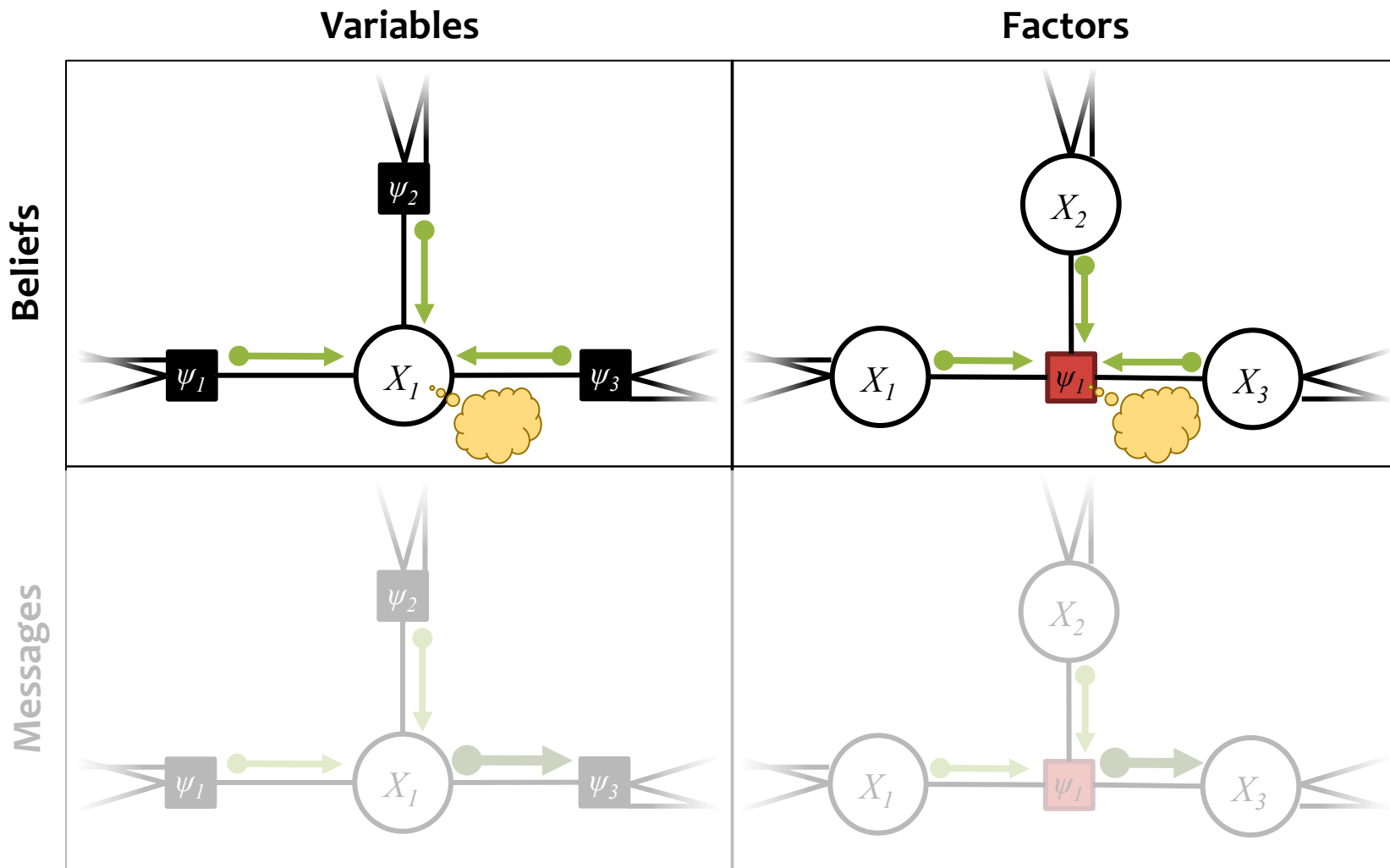
4. Compute the beliefs (unnormalized marginals).

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

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

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

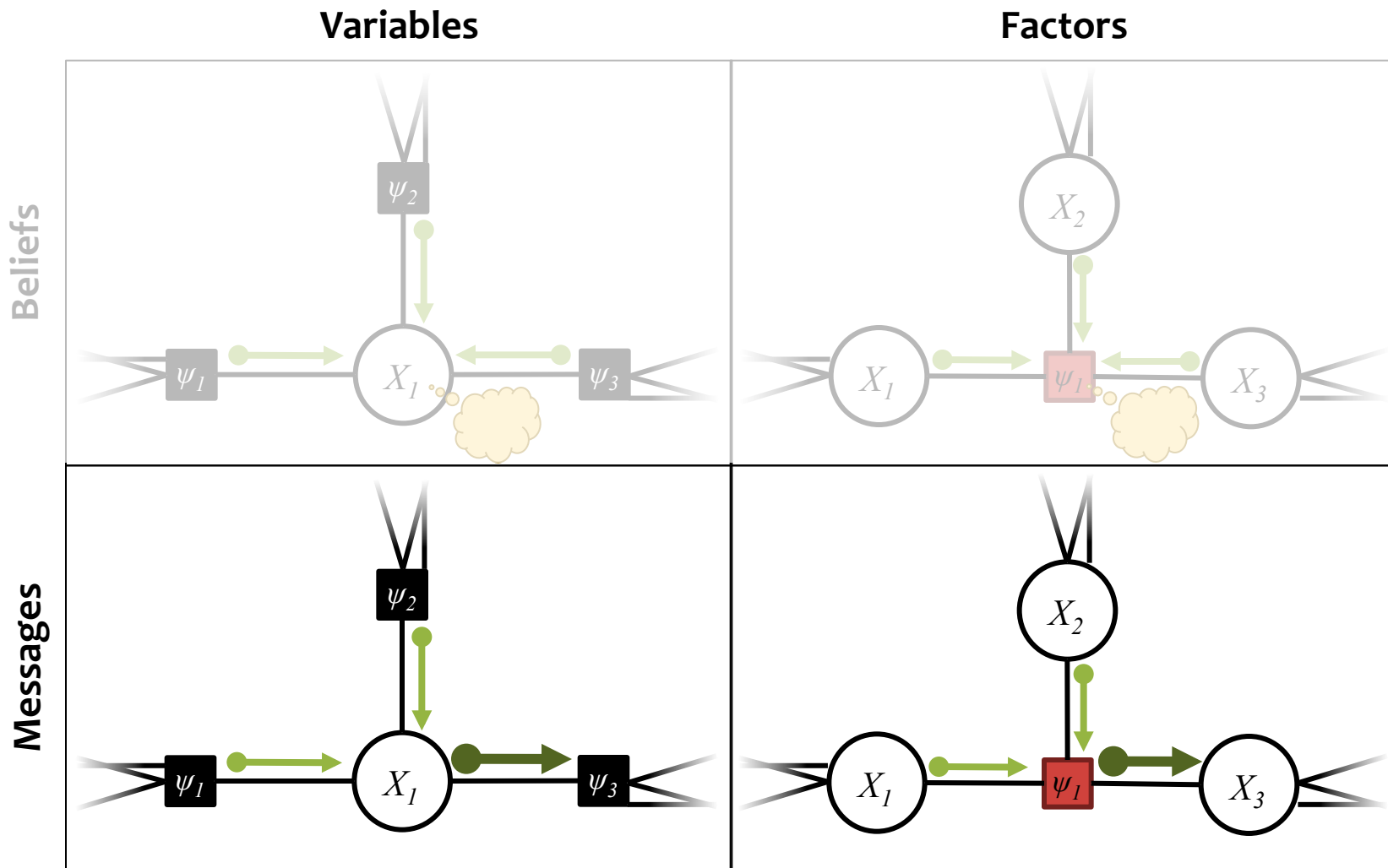
Sum-Product Belief Propagation



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

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

Sum-Product Belief Propagation



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

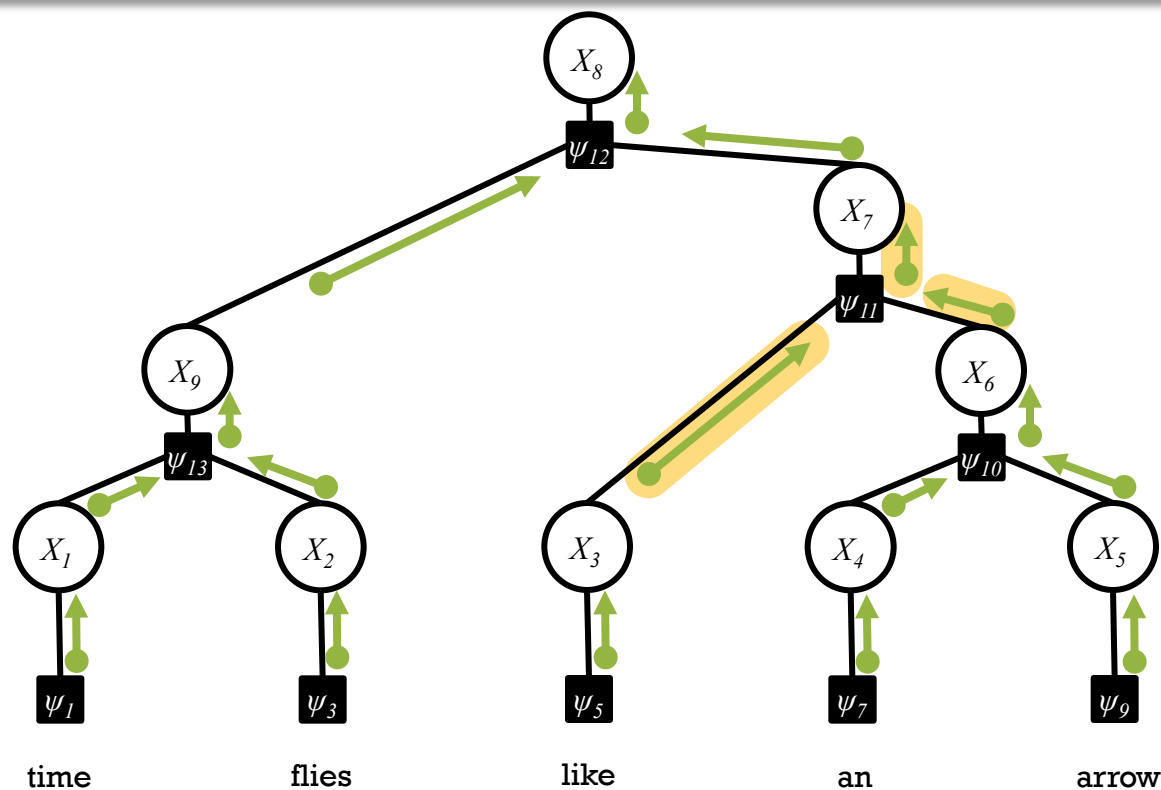
$$\mu_{\alpha \rightarrow i}(x_i) = \sum_{\mathbf{x}_{\alpha} : \mathbf{x}_{\alpha}[i] = x_i} \psi_{\alpha}(\mathbf{x}_{\alpha}) \prod_{j \in \mathcal{N}(\alpha) \setminus i} \mu_{j \rightarrow \alpha}(\mathbf{x}_{\alpha}[j])$$

(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.

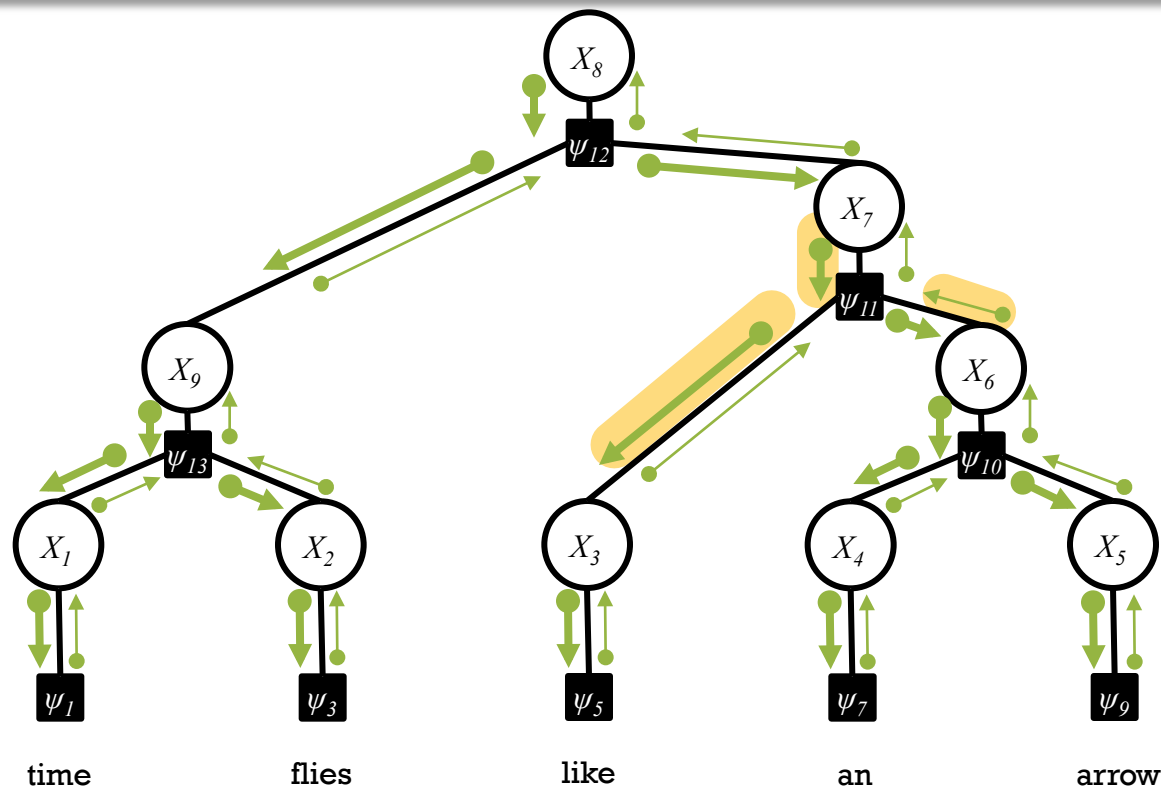


(Acyclic) Belief Propagation

In a factor graph with no cycles:

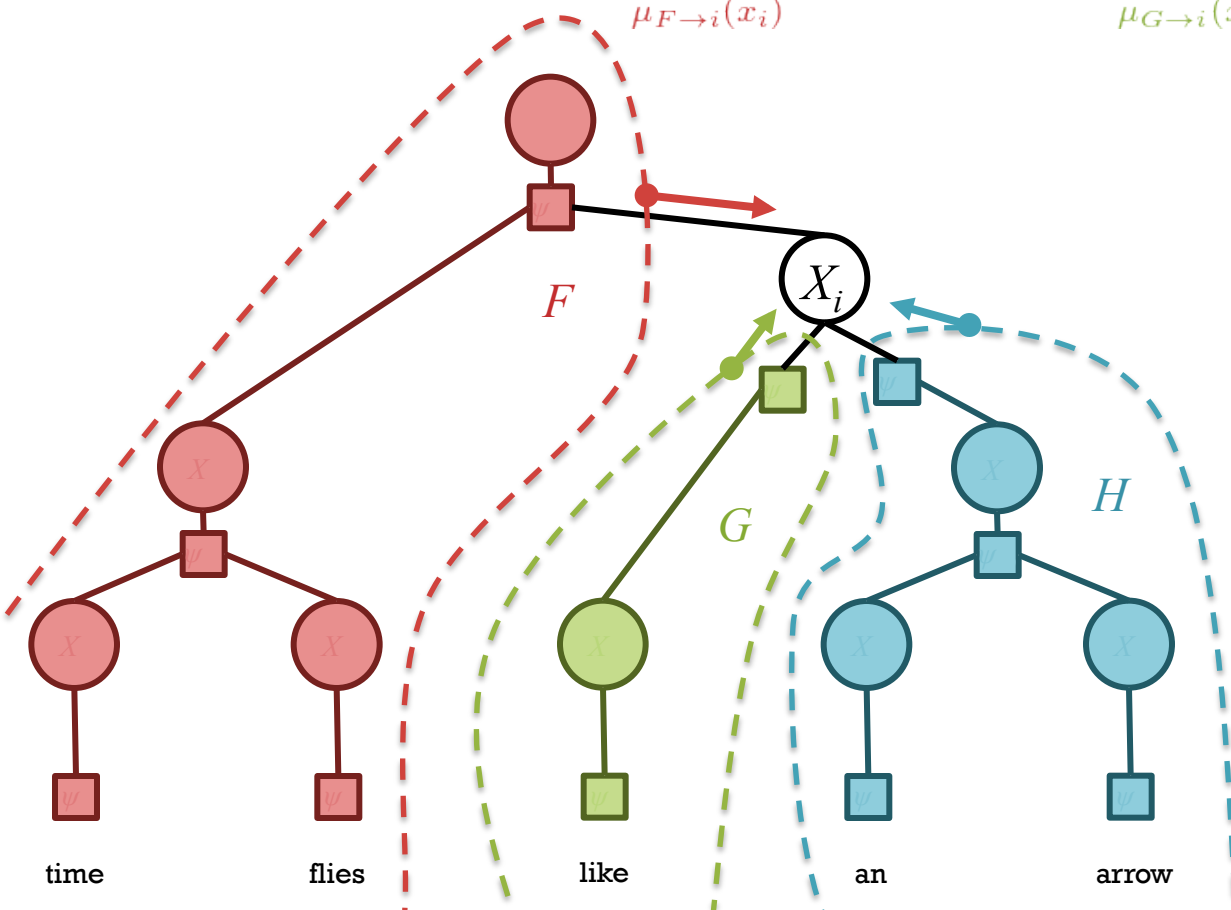
1. Pick any node to serve as the root.
2. Send messages from the **leaves** to the **root**.
3. Send messages from the **root** to the **leaves**.

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



Acyclic BP as Dynamic Programming

$$\begin{aligned}
 p(X_i = x_i) \propto b_i(x_i) &= \sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha} \psi_{\alpha}(\mathbf{x}_{\alpha}) \\
 &= \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq F} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{F \rightarrow i}(x_i)} \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq G} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{G \rightarrow i}(x_i)} \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq H} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{H \rightarrow i}(x_i)}
 \end{aligned}$$



Subproblem:

Inference using just the factors in subgraph H

Acyclic BP as Dynamic Programming

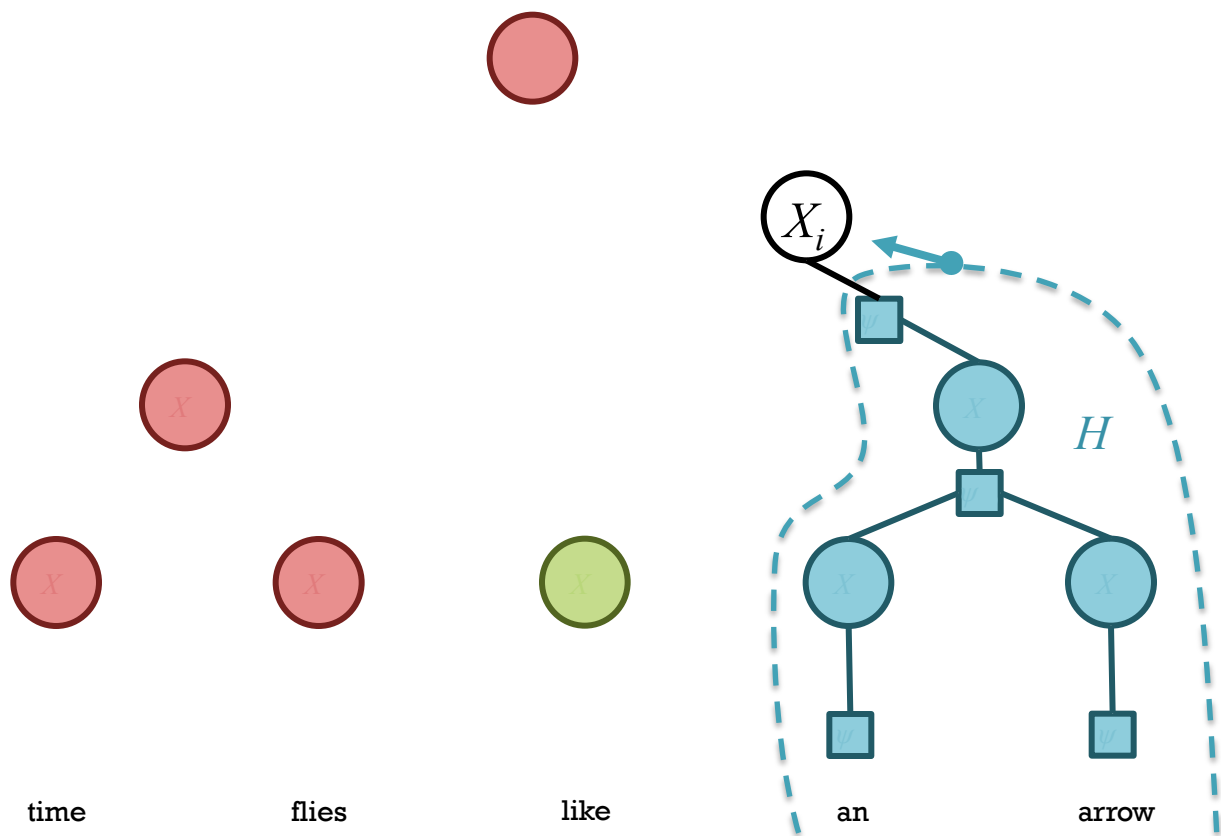
$$\begin{aligned}
 p(X_i = x_i) \propto b_i(x_i) &= \sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha} \psi_{\alpha}(\mathbf{x}_{\alpha}) \\
 &= \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq F} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{F \rightarrow i}(x_i)} \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq G} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{G \rightarrow i}(x_i)} \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq H} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{H \rightarrow i}(x_i)}
 \end{aligned}$$

Subproblem:

Inference using just the factors in subgraph H

The marginal of X_i in that smaller model is the message sent to X_i from subgraph H

Message to a variable



Acyclic BP as Dynamic Programming

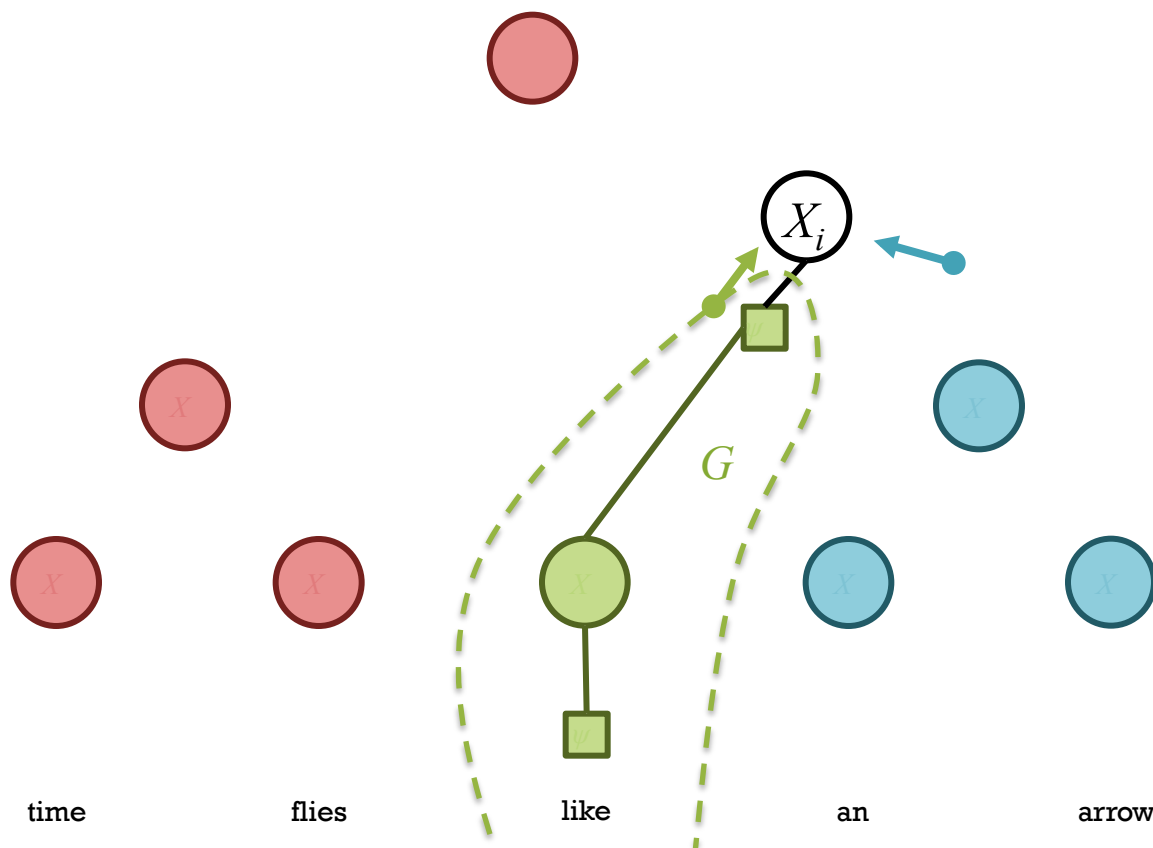
$$\begin{aligned}
 p(X_i = x_i) \propto b_i(x_i) &= \sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha} \psi_{\alpha}(\mathbf{x}_{\alpha}) \\
 &= \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq F} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{F \rightarrow i}(x_i)} \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq G} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{G \rightarrow i}(x_i)} \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq H} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{H \rightarrow i}(x_i)}
 \end{aligned}$$

Subproblem:

Inference using just the factors in subgraph H

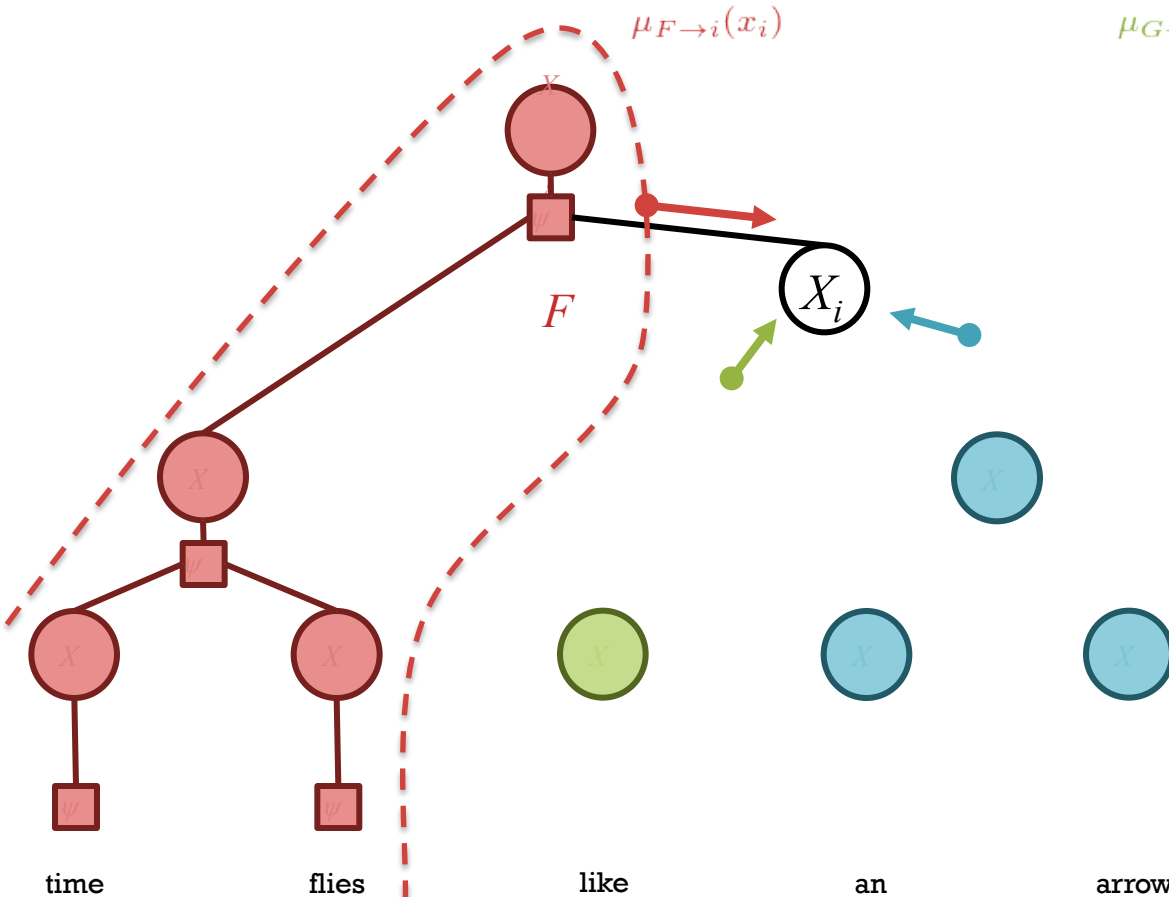
The marginal of X_i in that smaller model is the message sent to X_i from subgraph H

Message to a variable



Acyclic BP as Dynamic Programming

$$\begin{aligned}
 p(X_i = x_i) &\propto b_i(x_i) = \sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha} \psi_{\alpha}(\mathbf{x}_{\alpha}) \\
 &= \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq F} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{F \rightarrow i}(x_i)} \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq G} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{G \rightarrow i}(x_i)} \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq H} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{H \rightarrow i}(x_i)}
 \end{aligned}$$



Subproblem:

Inference using just the factors in subgraph H

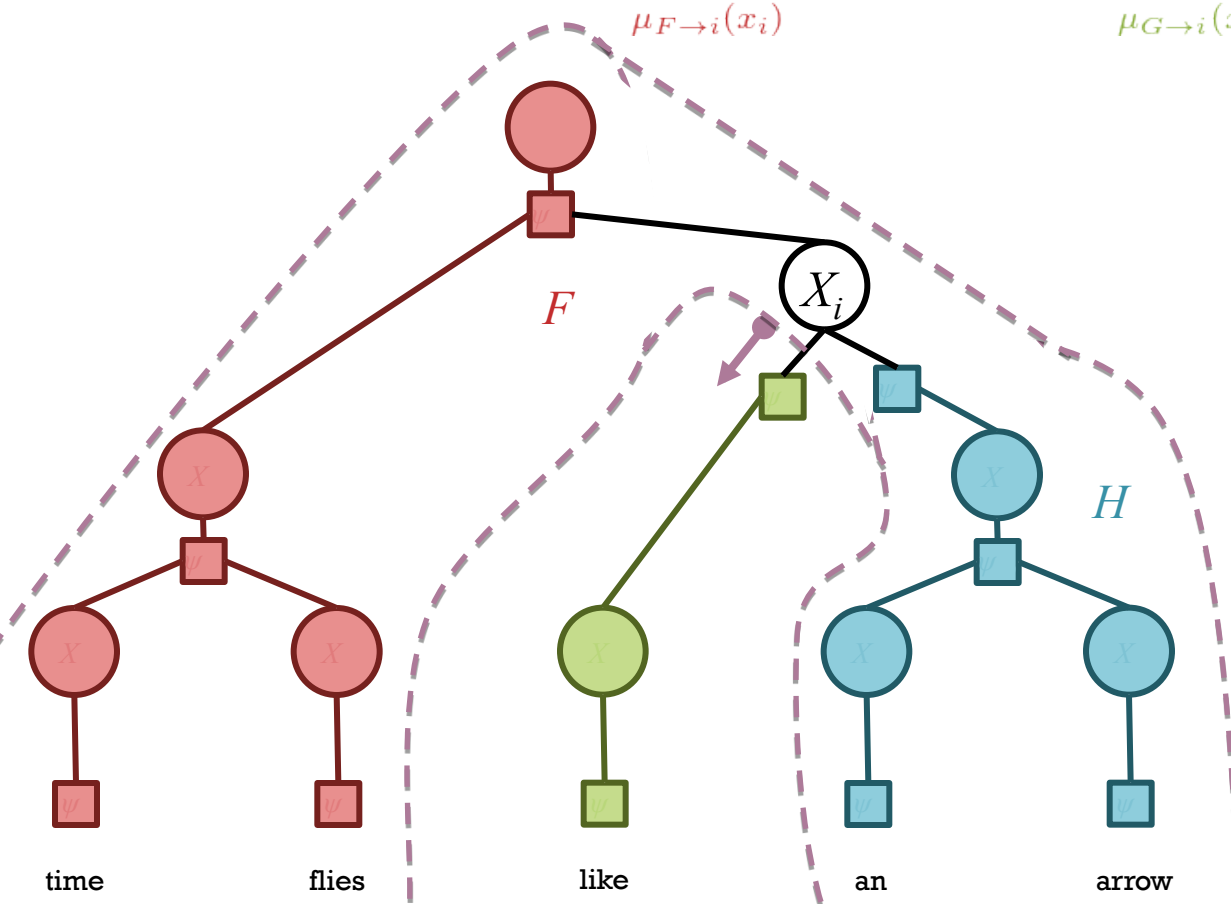
The marginal of X_i in that smaller model is the message sent to X_i from subgraph H

Message to a variable

Acyclic BP as Dynamic Programming

$$p(X_i = x_i) \propto b_i(x_i) = \sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha} \psi_{\alpha}(\mathbf{x}_{\alpha})$$

$$= \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq F} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{F \rightarrow i}(x_i)} \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq G} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{G \rightarrow i}(x_i)} \underbrace{\left(\sum_{\mathbf{x}: \mathbf{x}[i] = x_i} \prod_{\alpha \subseteq H} \psi_{\alpha}(\mathbf{x}_{\alpha}) \right)}_{\mu_{H \rightarrow i}(x_i)}$$



Subproblem:

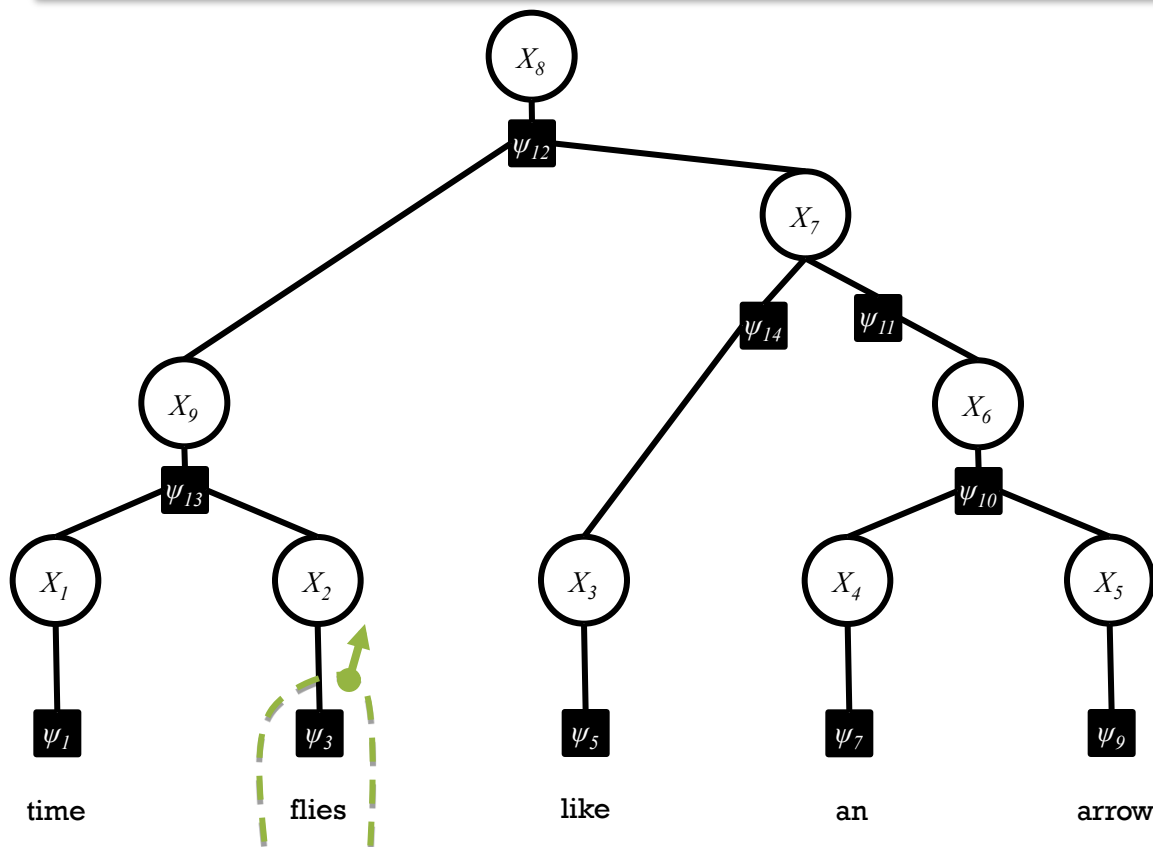
Inference using just the factors in subgraph $F \cup H$

The marginal of X_i in that smaller model is the message sent by X_i out of subgraph $F \cup H$

Message from a variable

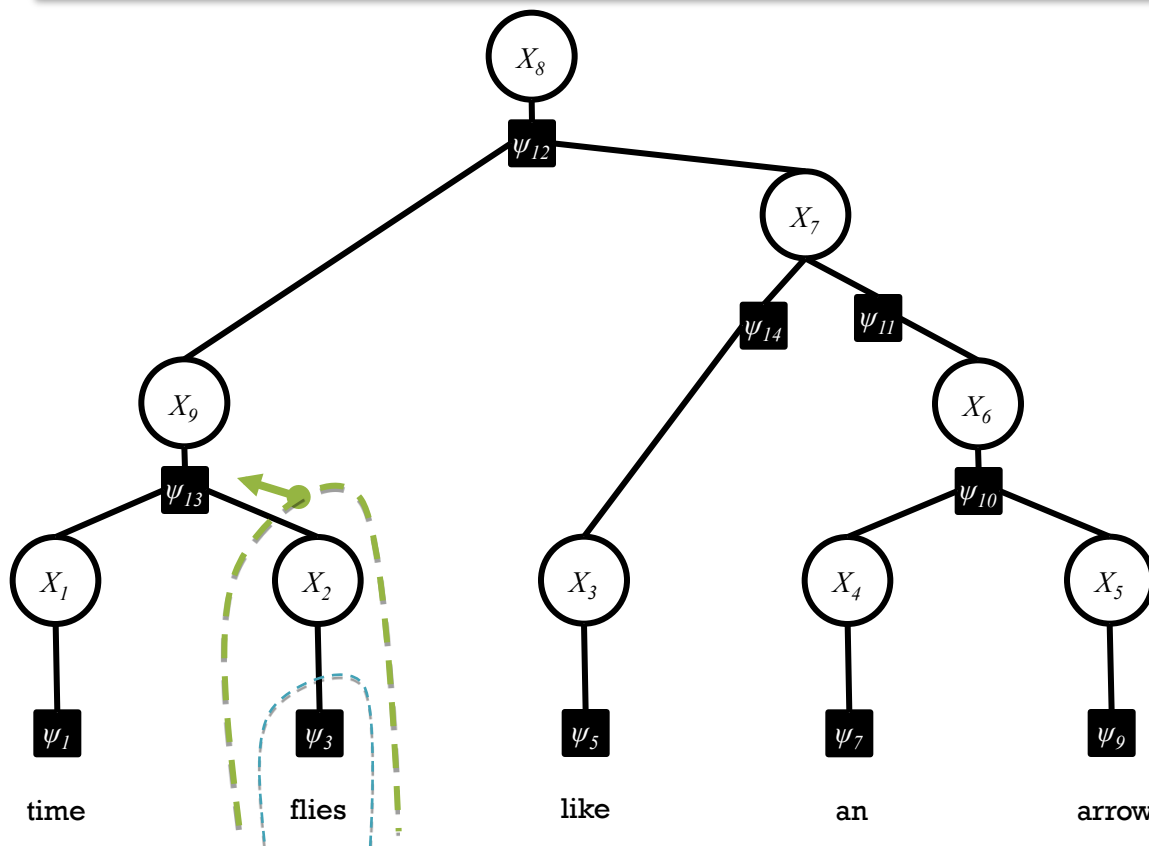
Acyclic BP as Dynamic Programming

- If you want the **marginal** $p_i(x_i)$ where X_i has degree k , you can think of that summation as a **product of k marginals** computed on smaller subgraphs.
- Each subgraph is obtained by **cutting** some edge of the tree.
- The message-passing algorithm uses **dynamic programming** to compute the marginals on all such subgraphs, working from **smaller to bigger**. So you can compute all the marginals.



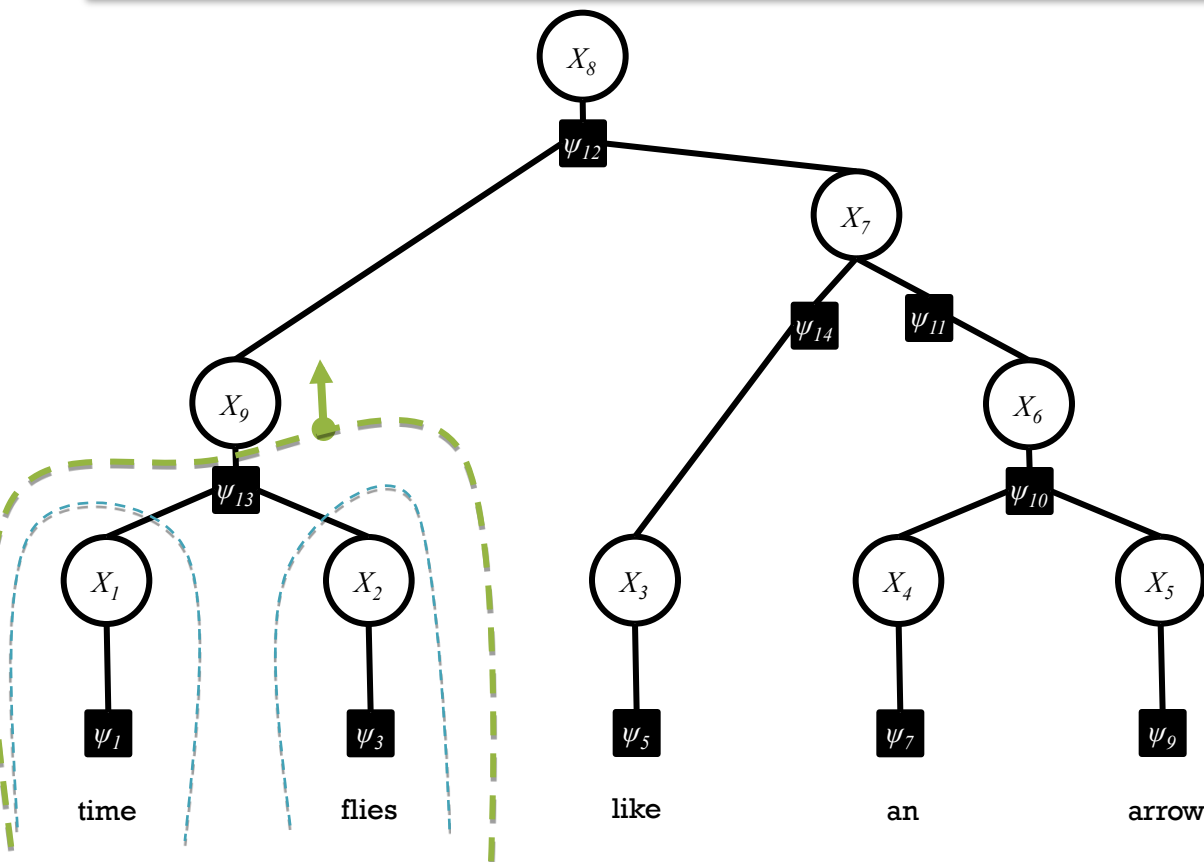
Acyclic BP as Dynamic Programming

- If you want the **marginal** $p_i(x_i)$ where X_i has degree k , you can think of that summation as a **product of k marginals** computed on smaller subgraphs.
- Each subgraph is obtained by **cutting** some edge of the tree.
- The message-passing algorithm uses **dynamic programming** to compute the marginals on all such subgraphs, working from **smaller to bigger**. So you can compute all the marginals.



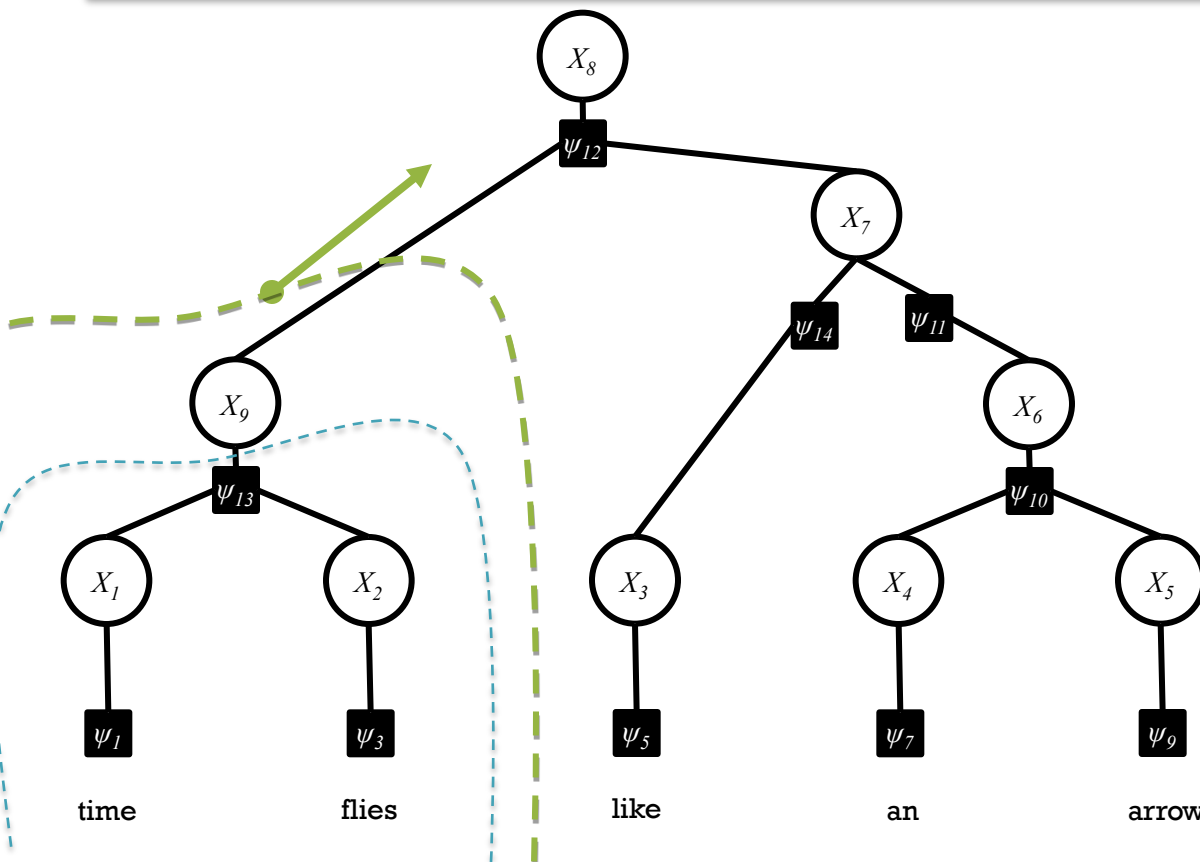
Acyclic BP as Dynamic Programming

- If you want the **marginal** $p_i(x_i)$ where X_i has degree k , you can think of that summation as a **product of k marginals** computed on smaller subgraphs.
- Each subgraph is obtained by **cutting** some edge of the tree.
- The message-passing algorithm uses **dynamic programming** to compute the marginals on all such subgraphs, working from **smaller to bigger**. So you can compute all the marginals.



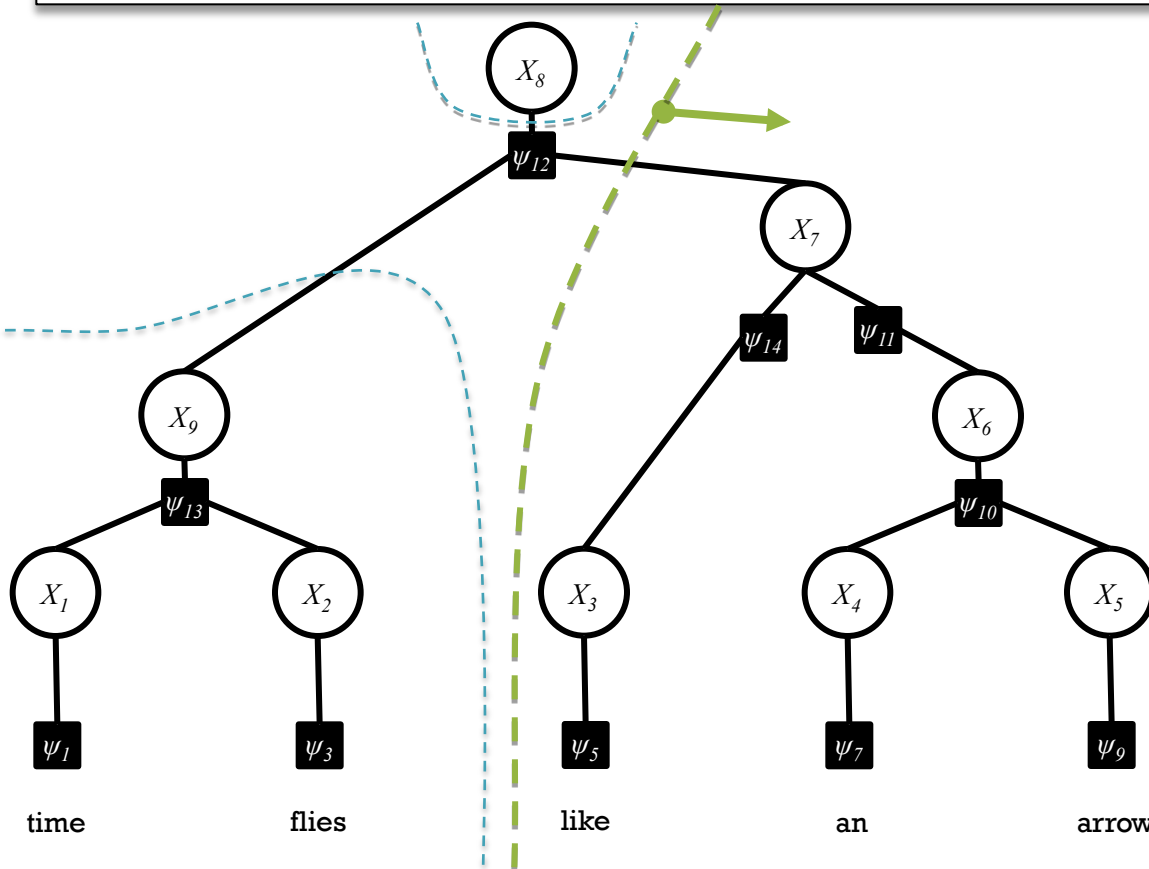
Acyclic BP as Dynamic Programming

- If you want the **marginal** $p_i(x_i)$ where X_i has degree k , you can think of that summation as a **product of k marginals** computed on smaller subgraphs.
- Each subgraph is obtained by **cutting** some edge of the tree.
- The message-passing algorithm uses **dynamic programming** to compute the marginals on all such subgraphs, working from **smaller to bigger**. So you can compute all the marginals.



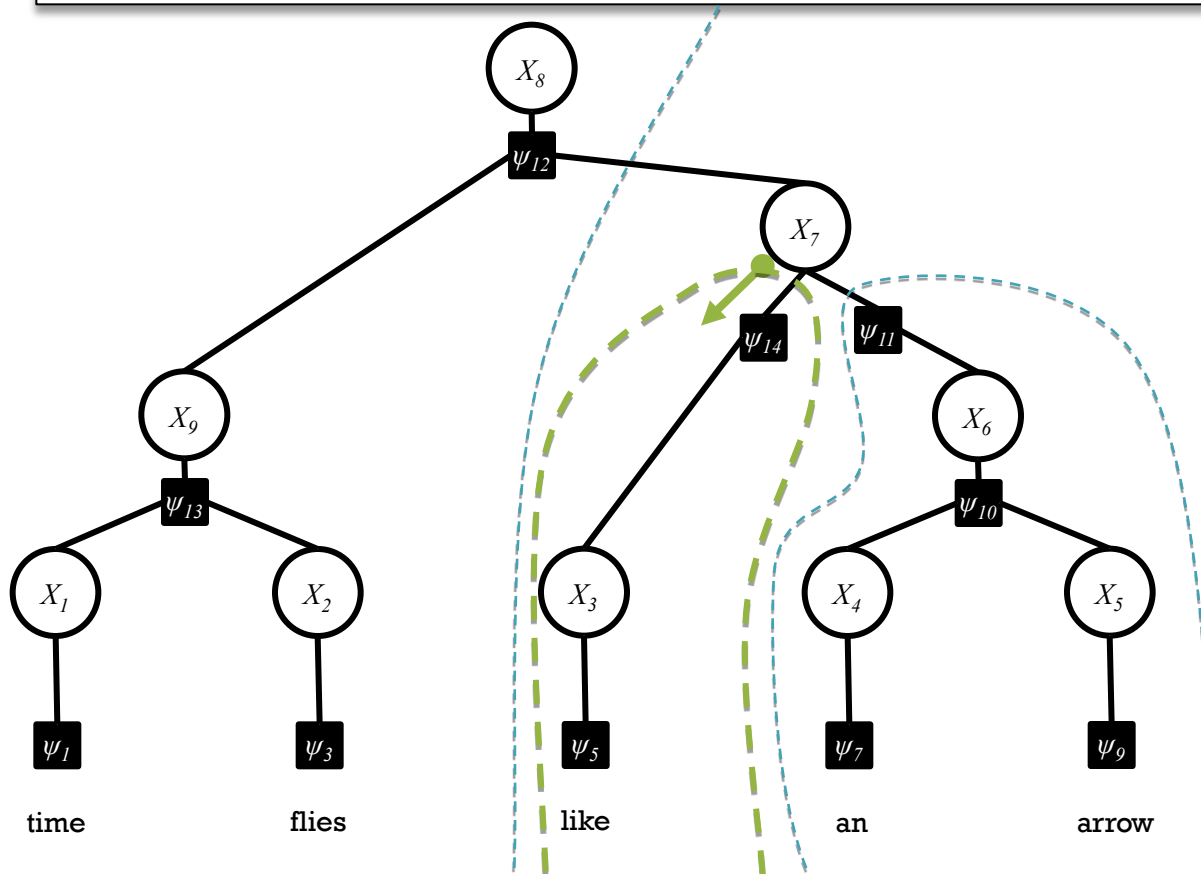
Acyclic BP as Dynamic Programming

- If you want the **marginal** $p_i(x_i)$ where X_i has degree k , you can think of that summation as a **product of k marginals** computed on smaller subgraphs.
- Each subgraph is obtained by **cutting** some edge of the tree.
- The message-passing algorithm uses **dynamic programming** to compute the marginals on all such subgraphs, working from **smaller to bigger**. So you can compute all the marginals.



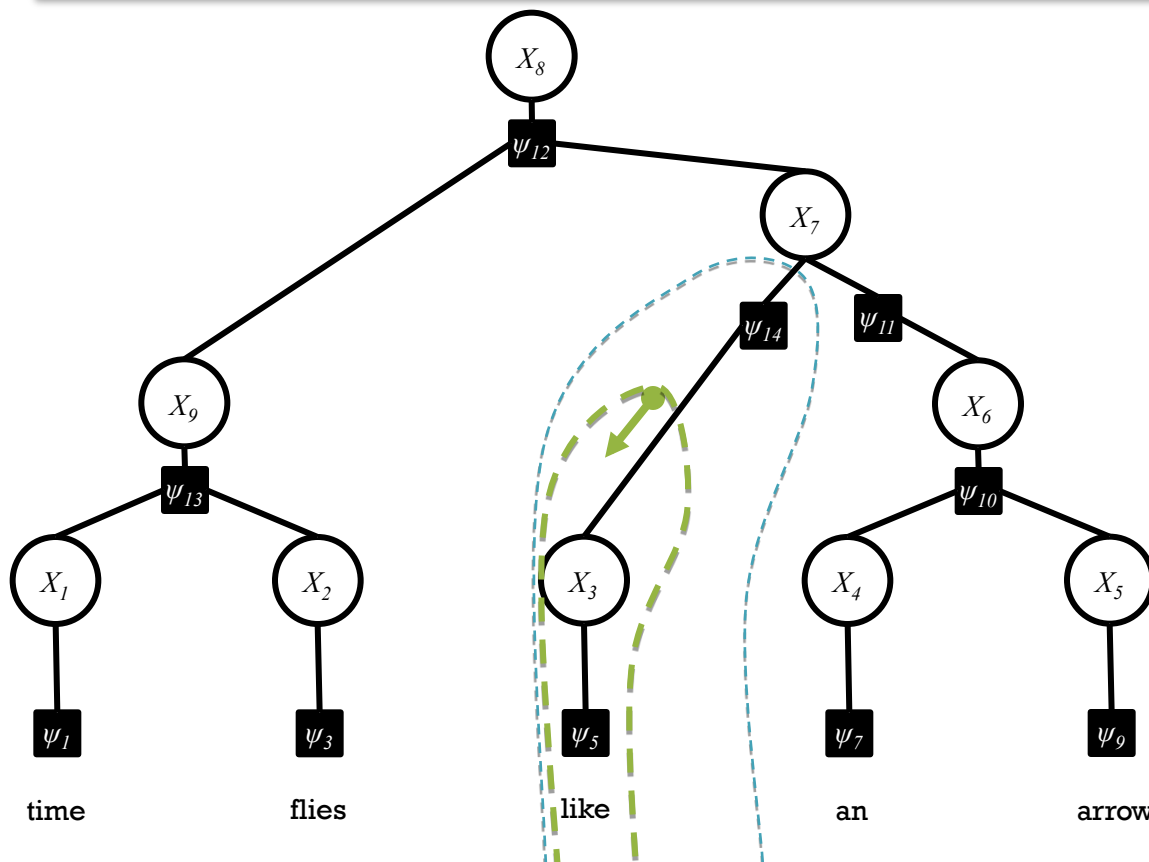
Acyclic BP as Dynamic Programming

- If you want the **marginal** $p_i(x_i)$ where X_i has degree k , you can think of that summation as a **product of k marginals** computed on smaller subgraphs.
- Each subgraph is obtained by **cutting** some edge of the tree.
- The message-passing algorithm uses **dynamic programming** to compute the marginals on all such subgraphs, working from **smaller to bigger**. So you can compute all the marginals.



Acyclic BP as Dynamic Programming

- If you want the **marginal** $p_i(x_i)$ where X_i has degree k , you can think of that summation as a **product of k marginals** computed on smaller subgraphs.
- Each subgraph is obtained by **cutting** some edge of the tree.
- The message-passing algorithm uses **dynamic programming** to compute the marginals on all such subgraphs, working from **smaller to bigger**. So you can compute all the marginals.



Exact MAP inference for factor trees

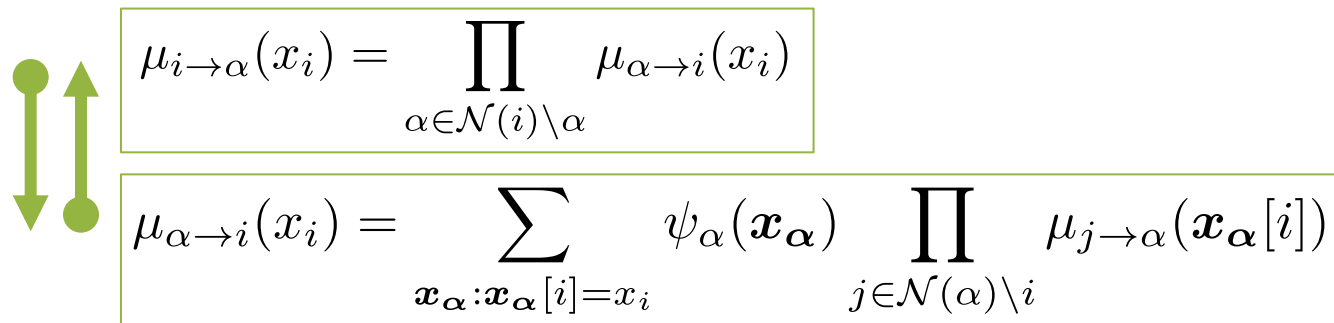
MAX-PRODUCT BELIEF PROPAGATION

Max-product Belief Propagation

- **Sum-product BP** can be used to compute the marginals, $p_i(X_i)$
- **Max-product BP** can be used to compute the most likely assignment, $X^* = \operatorname{argmax}_X p(X)$

Max-product Belief Propagation

- Change the sum to a max:



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

- **Max-product BP** computes **max-marginals**
 - The max-marginal $b_i(x_i)$ is the (unnormalized) probability of the MAP assignment under the constraint $X_i = x_i$.
 - For an acyclic graph, the MAP assignment (assuming there are no ties) is given by:

$$x_i^* = \arg \max_{x_i} b_i(x_i)$$

Max-product Belief Propagation

- Change the sum to a max:


$$\mu_{i \rightarrow \alpha}(x_i) = \prod_{\alpha \in \mathcal{N}(i) \setminus \alpha} \mu_{\alpha \rightarrow i}(x_i)$$
$$\mu_{\alpha \rightarrow i}(x_i) = \max_{\mathbf{x}_\alpha : \mathbf{x}_\alpha[i] = x_i} \psi_\alpha(\mathbf{x}_\alpha) \prod_{j \in \mathcal{N}(\alpha) \setminus i} \mu_{j \rightarrow \alpha}(\mathbf{x}_\alpha[j])$$

- **Max-product BP** computes **max-marginals**
 - The max-marginal $b_i(x_i)$ is the (unnormalized) probability of the MAP assignment under the constraint $X_i = x_i$.
 - For an acyclic graph, the MAP assignment (assuming there are no ties) is given by:

$$x_i^* = \arg \max_{x_i} b_i(x_i)$$

Semirings

- Sum-product $+/*$ and max-product $\max/*$ are commutative semirings
- We can run BP with any such commutative semiring

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

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

- In practice, multiplying many small numbers together can yield underflow
 - instead of using $+/*$, we use log-add/+
 - Instead of using $\max/*$, we use $\max/+$

Summary

1. Factor Graphs

- Alternative representation of directed / undirected graphical models
- Make the cliques of an undirected GM explicit

2. Variable Elimination

- Simple and general approach to exact inference
- Just a matter of being clever when computing sum-products

3. Sum-product Belief Propagation

- Computes all the marginals and the partition function in only twice the work of Variable Elimination

4. Max-product Belief Propagation

- Identical to sum-product BP, but changes the semiring
- Computes: max-marginals, probability of MAP assignment, and (with backpointers) the MAP assignment itself.