School of Computer Science
Carnegie Mellon

10-708 Probabilistic Graphical Models

Exact Inference

Readings:

Jordan Chap. 3

Jordan Chap. 4

Matt Gormley

Lecture 7

February 3, 2016

Some slides and figures from Gormley & Eisner (2014, 2015) "Structured BP for NLP" tutorial

Housekeeping

- **Office Hours**

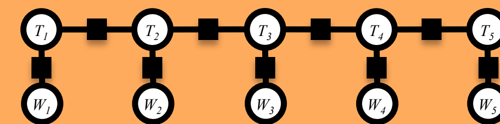
Exact Inference

1. Data

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

2. Model

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



3. Objective

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

5. Inference

1. Marginal Inference

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

2. Partition Function

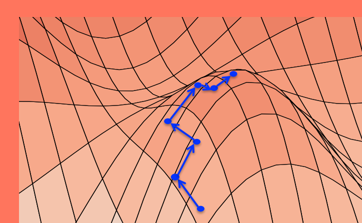
$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference

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

4. Learning

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



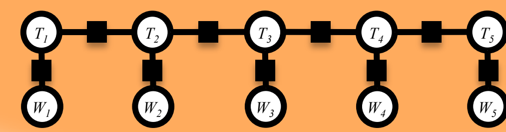
Goals for Today's Lecture

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**

1. Data

2. Model

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



3. Objective

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

5. Inference

1. Marginal Inference

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

2. Partition Function

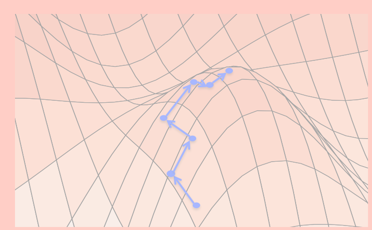
$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference

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

4. Learning

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

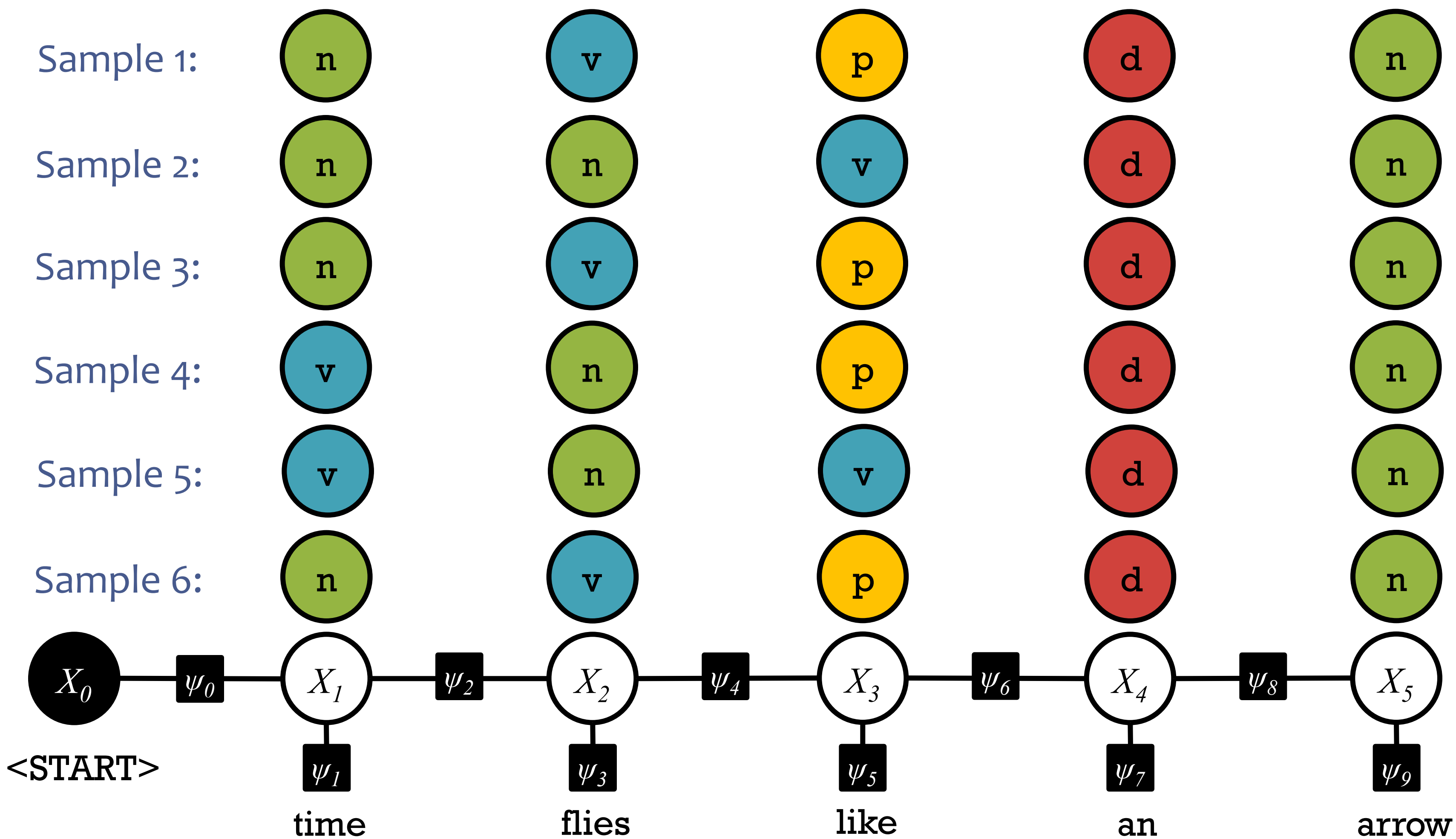


Representation of both directed and undirected graphical models

FACTOR GRAPHS

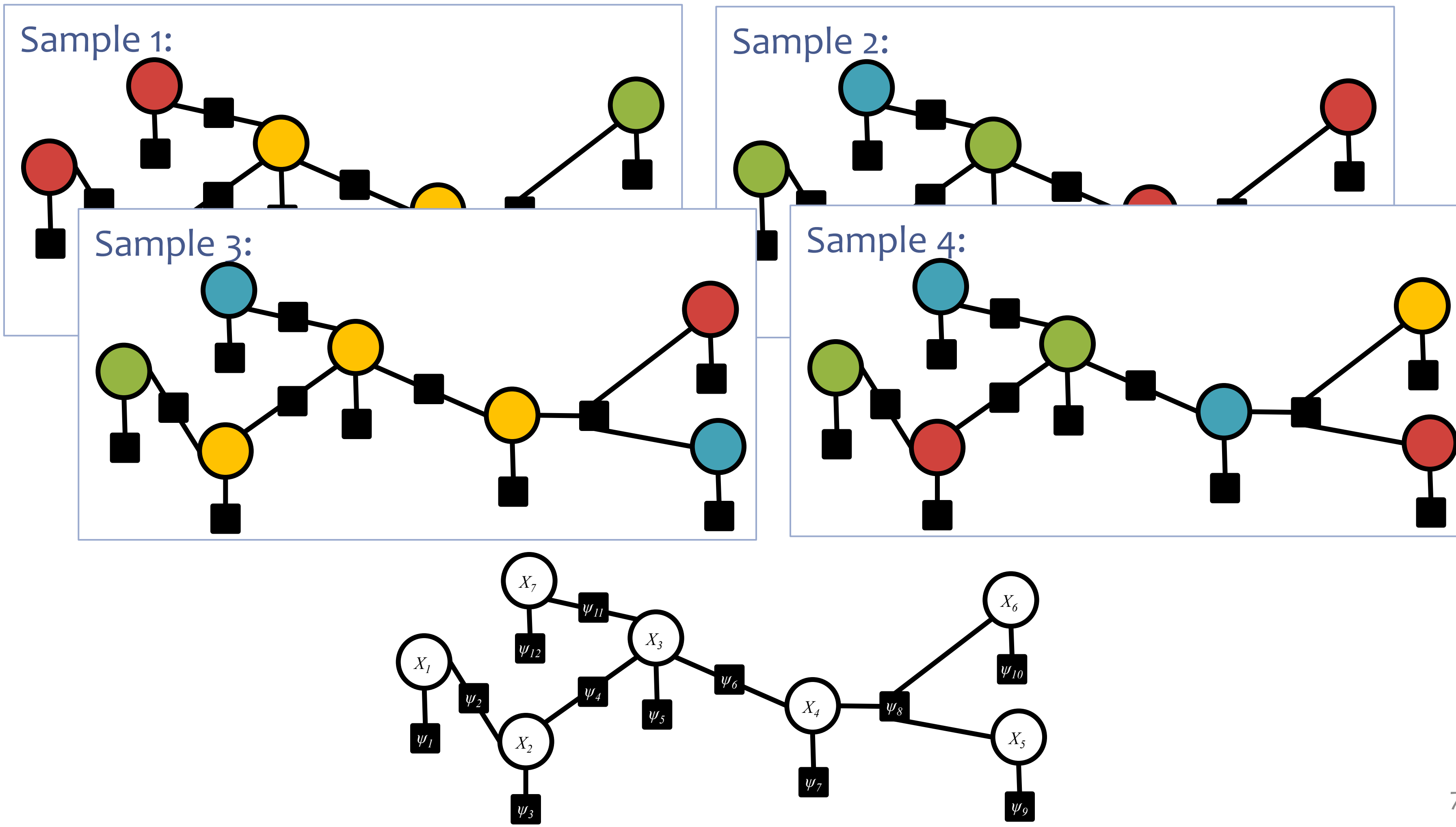
Sampling from a Joint Distribution

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



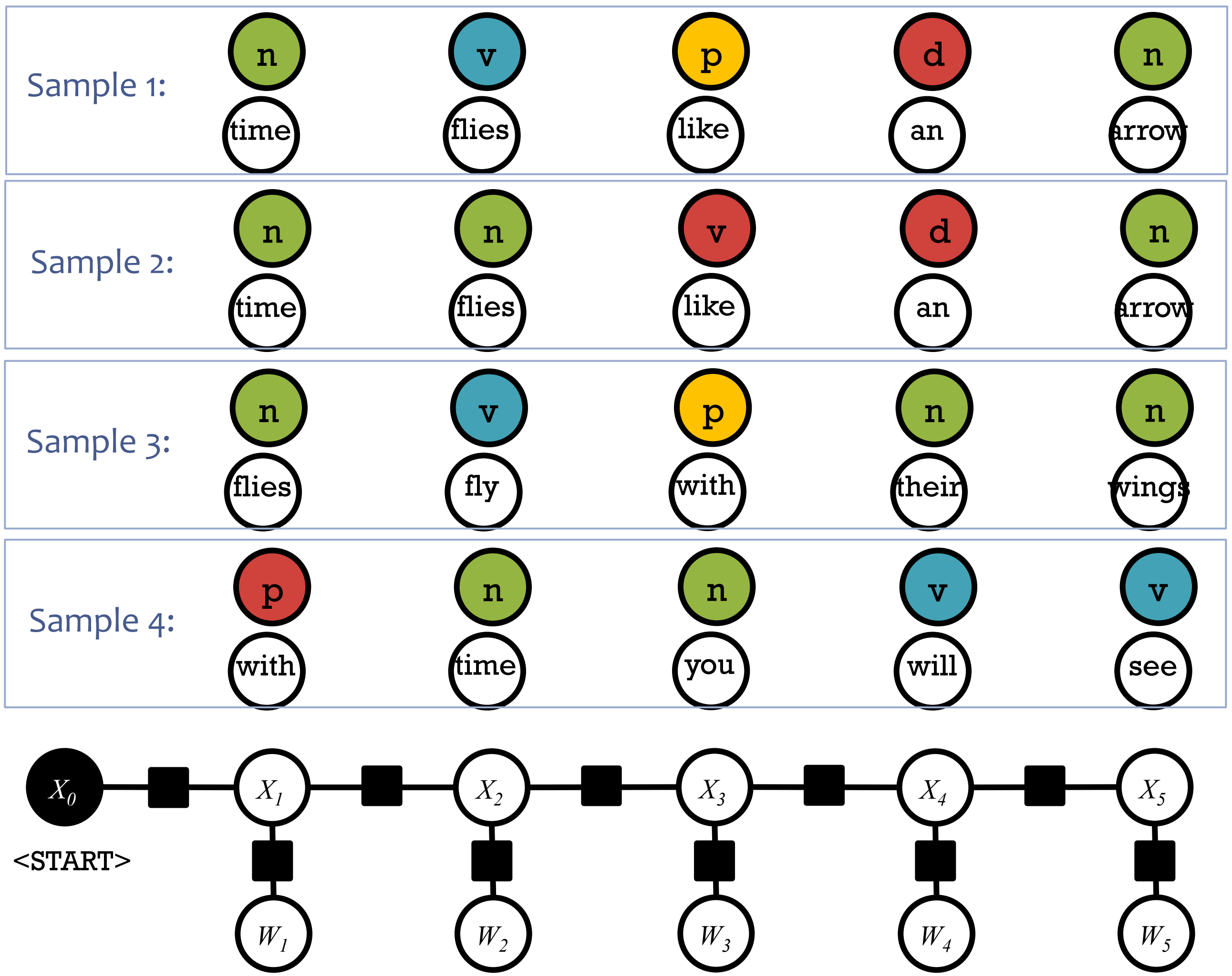
Sampling from a Joint Distribution

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



Sampling from a Joint Distribution

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



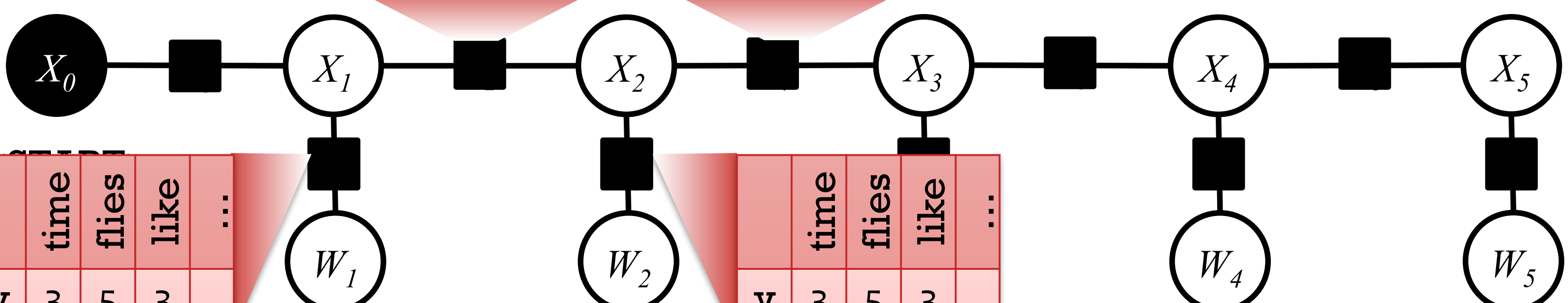
Factors have local opinions (≥ 0)

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.

	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



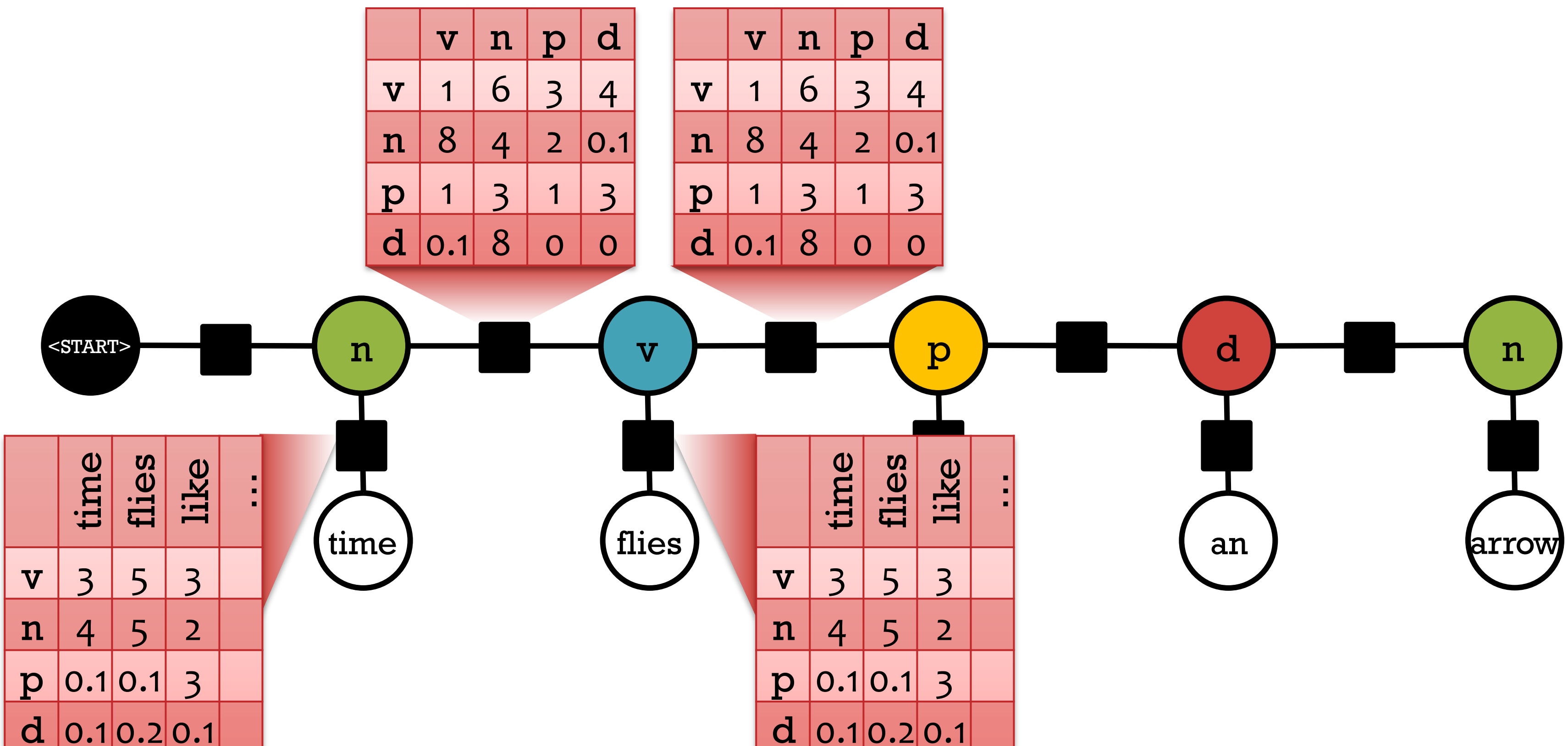
	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

Factors have local opinions (≥ 0)

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

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



	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

	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

Global probability = product of local opinions

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

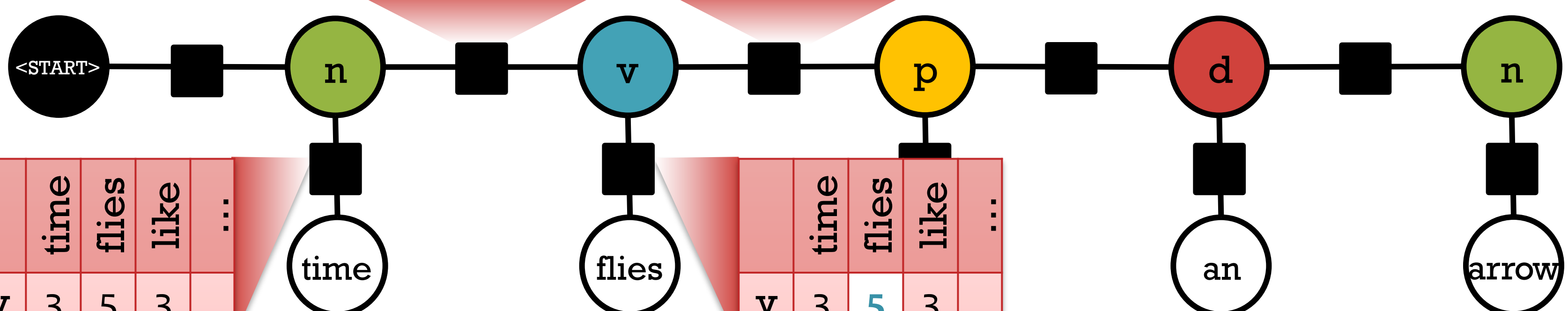
$$p(\text{n, v, p, d, n, time, flies, like, an, arrow}) = \frac{1}{Z}(4 * 8 * 5 * 3 * \ldots)$$

	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.



	time	flies	like	...
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

	time	flies	like	...
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

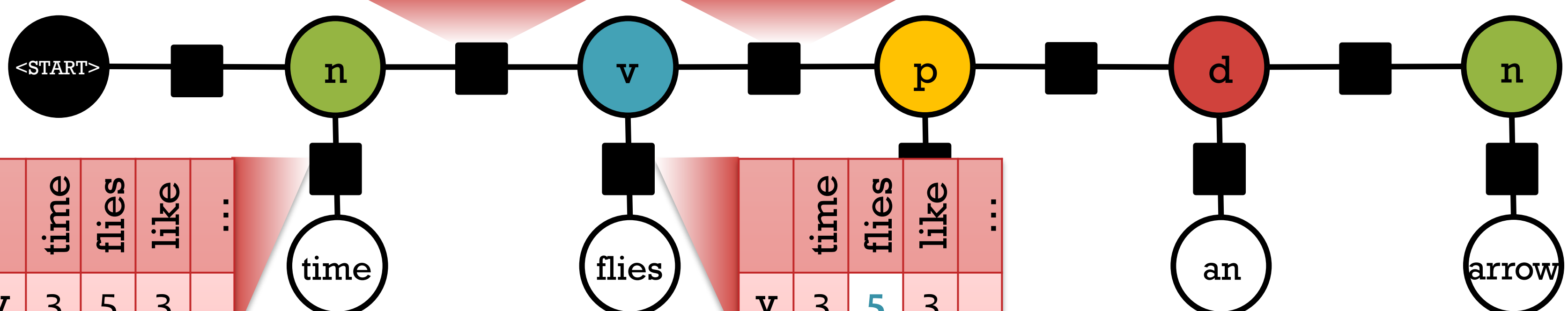
Markov Random Field (MRF)

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

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

	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



	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

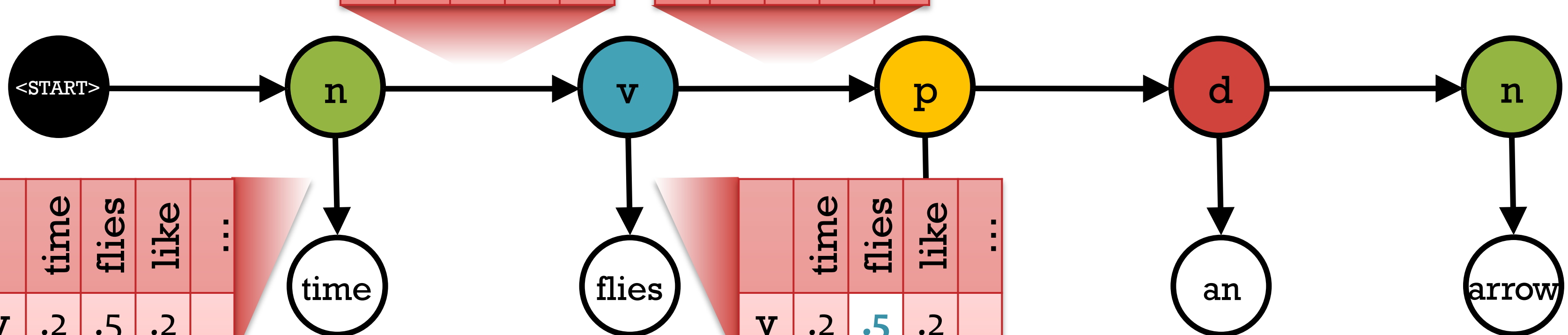
Bayesian Networks

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

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

	v	n	p	d
v	.1	.4	.2	.3
n	.8	.1	.1	0
p	.2	.3	.2	.3
d	.2	.8	0	0

	v	n	p	d
v	.1	.4	.2	.3
n	.8	.1	.1	0
p	.2	.3	.2	.3
d	.2	.8	0	0



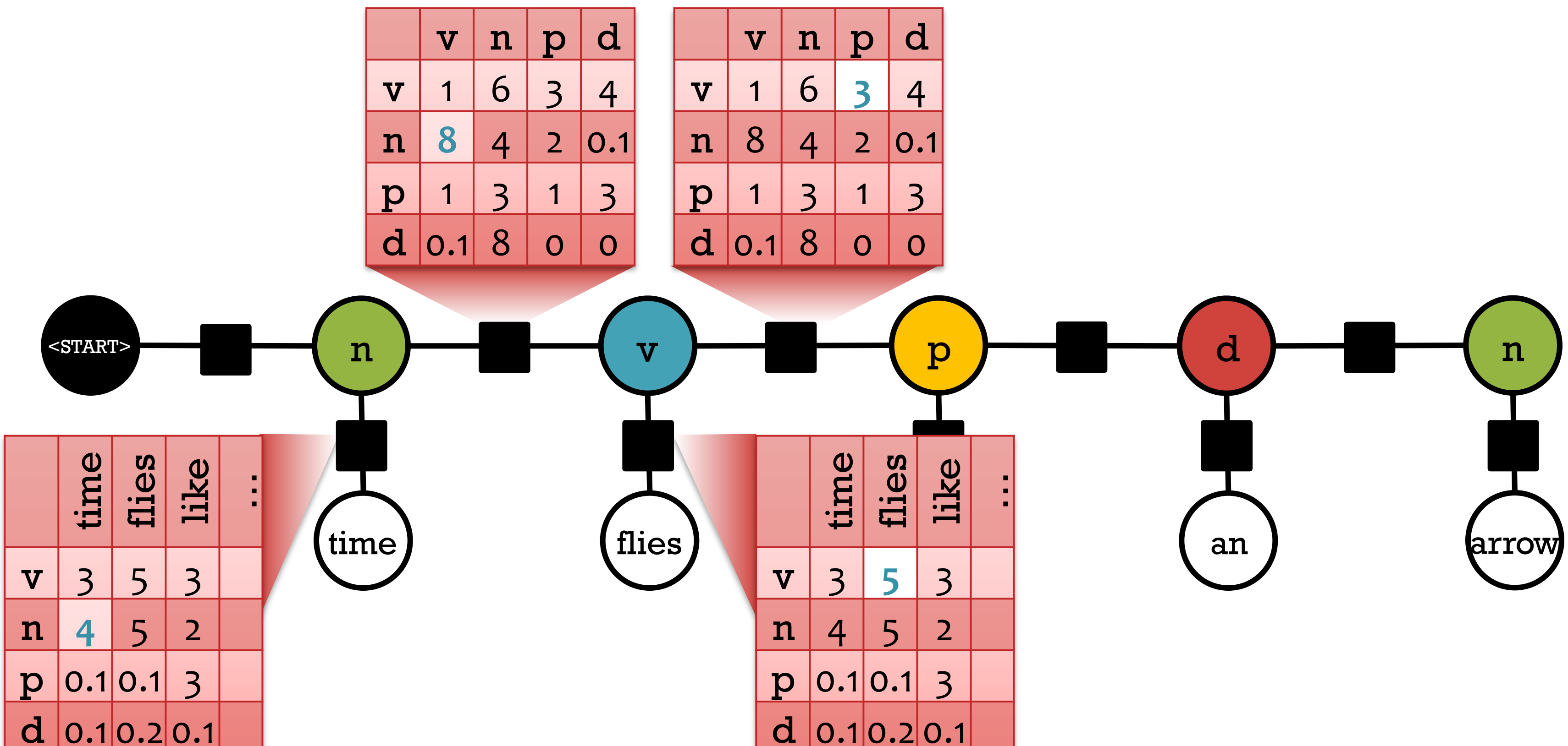
	time	flies	like	…
v	.2	.5	.2	
n	.3	.4	.2	
p	.1	.1	.3	
d	.1	.2	.1	

	time	flies	like	…
v	.2	.5	.2	
n	.3	.4	.2	
p	.1	.1	.3	
d	.1	.2	.1	

Markov Random Field (MRF)

Joint distribution over tags X_i <u>and</u> words W_i

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



	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

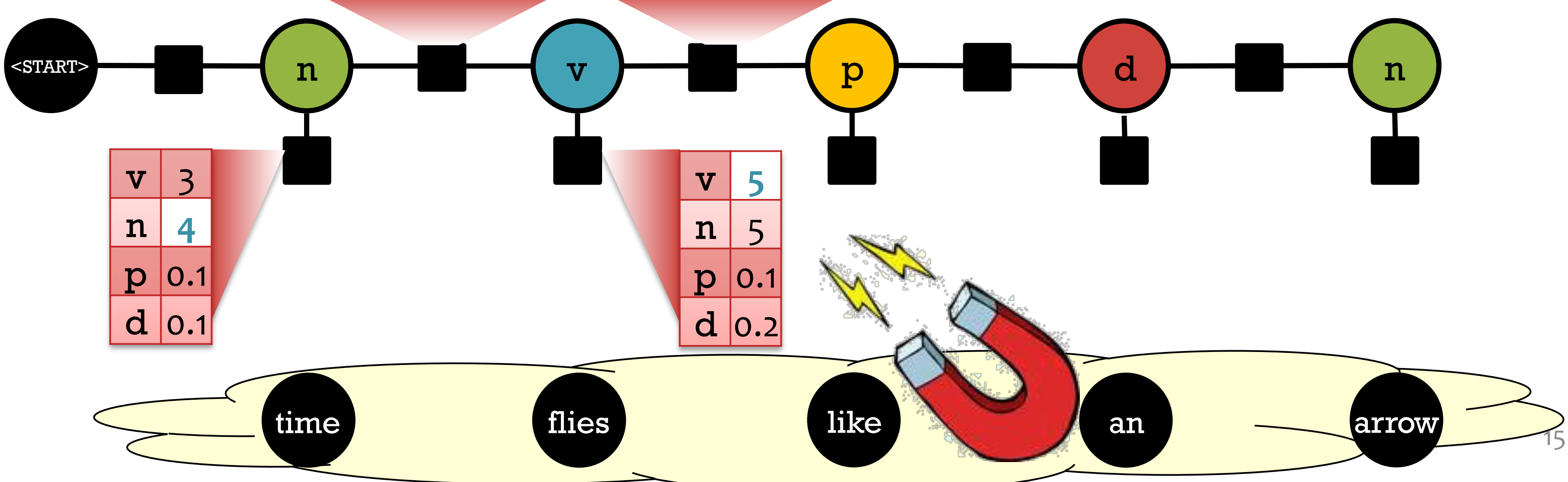
	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

	time	flies	like	…
v	3	5	3	
n	4	5	2	
p	0.1	0.1	3	
d	0.1	0.2	0.1	

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.

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



	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

v	3
n	4
p	0.1
d	0.1

v	5
n	5
p	0.1
d	0.2

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:

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

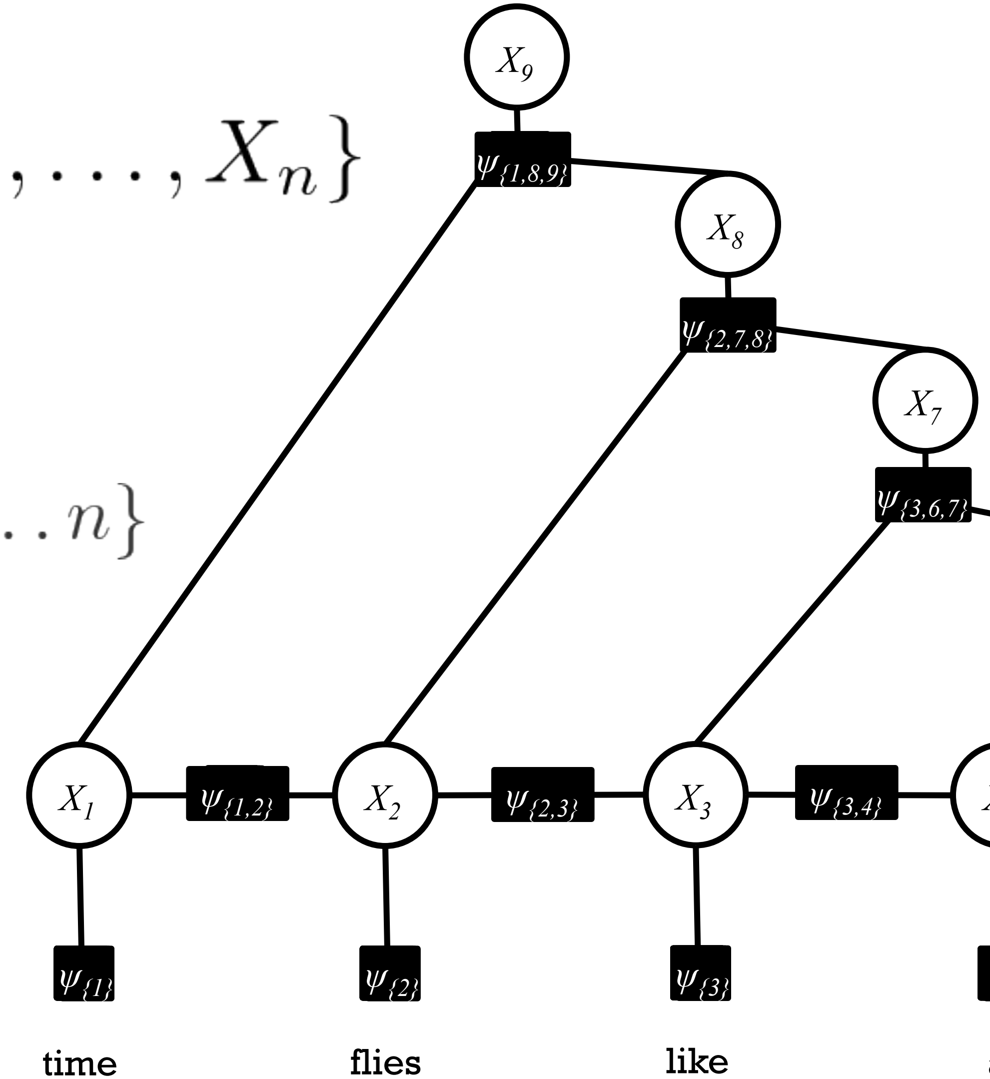
- Factors:

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

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

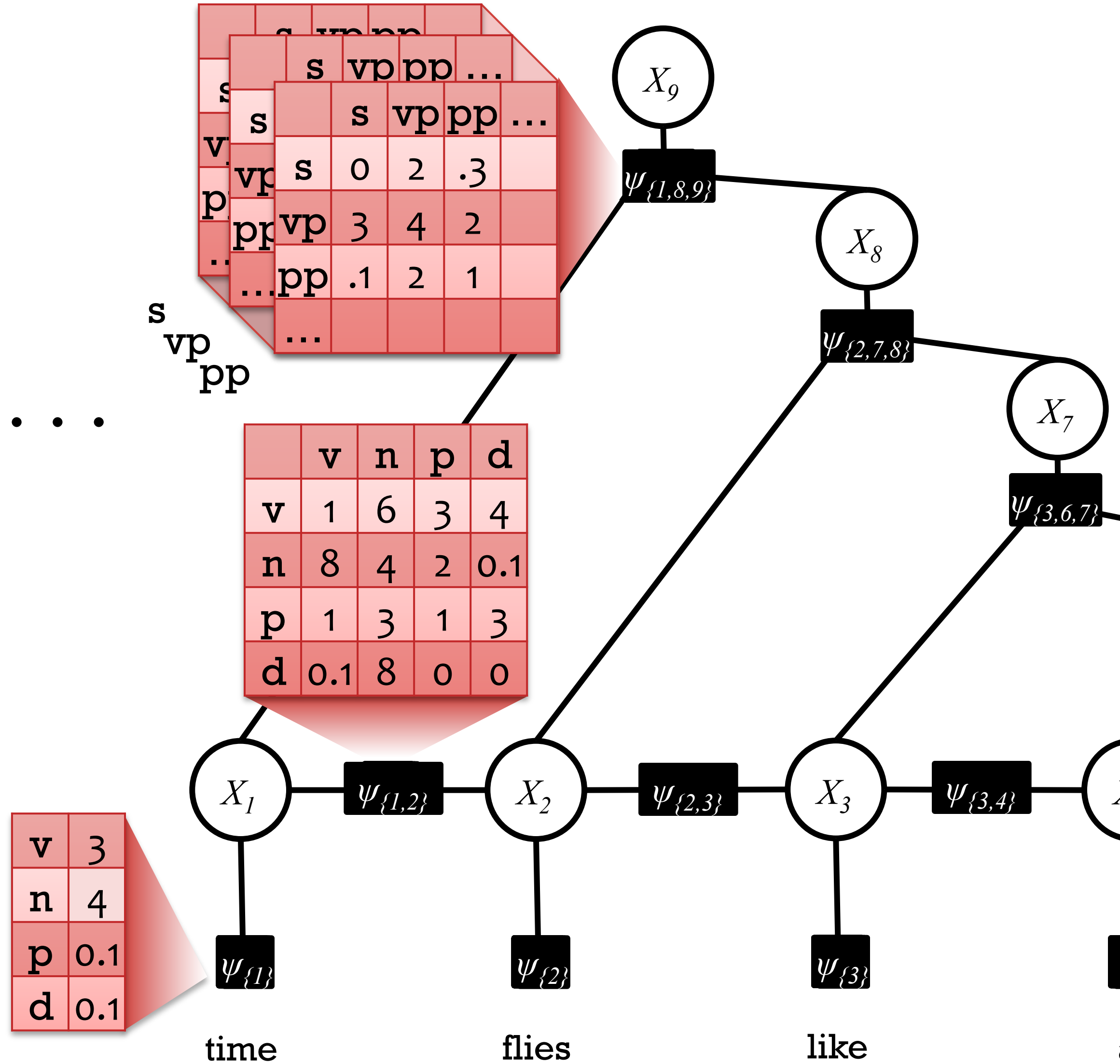
Joint Distribution

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



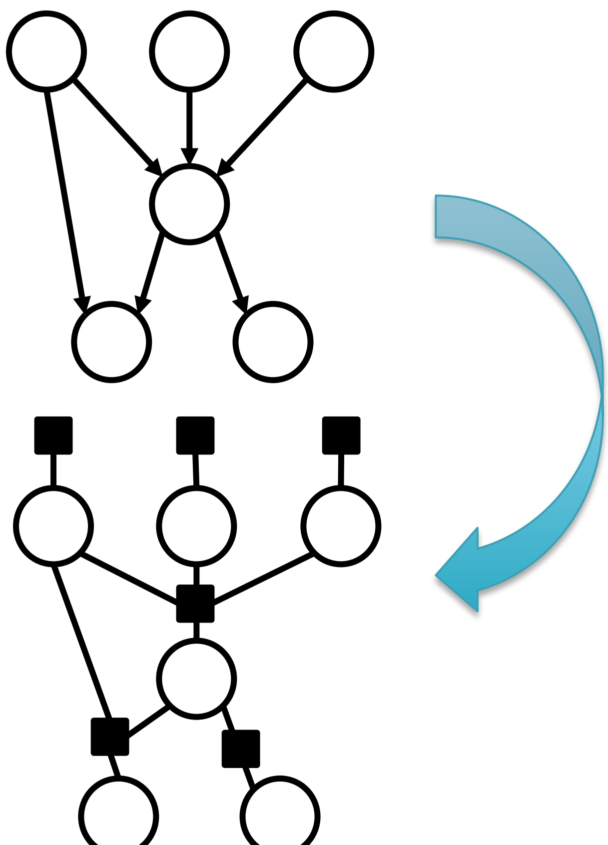
Factors are Tensors

- Factors:
  $$\psi_\alpha, \psi_\beta, \psi_\gamma, \ldots$$

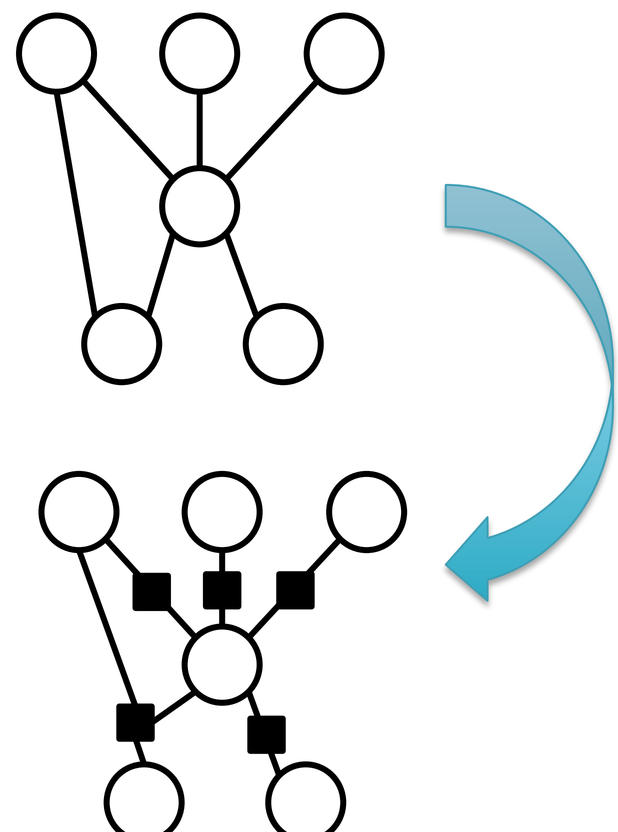


Converting to Factor Graphs

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



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

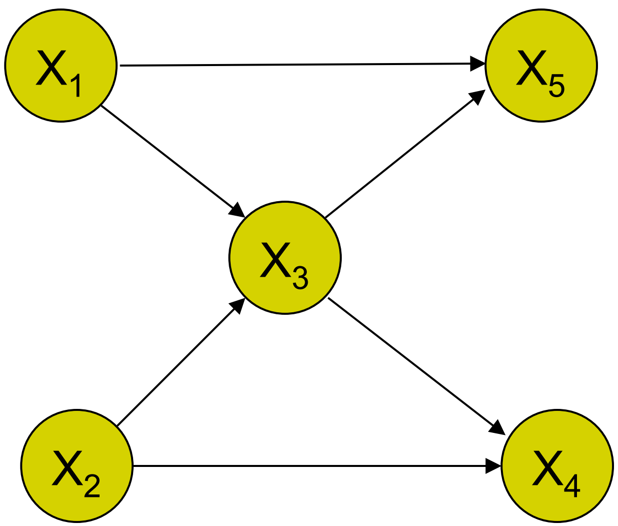


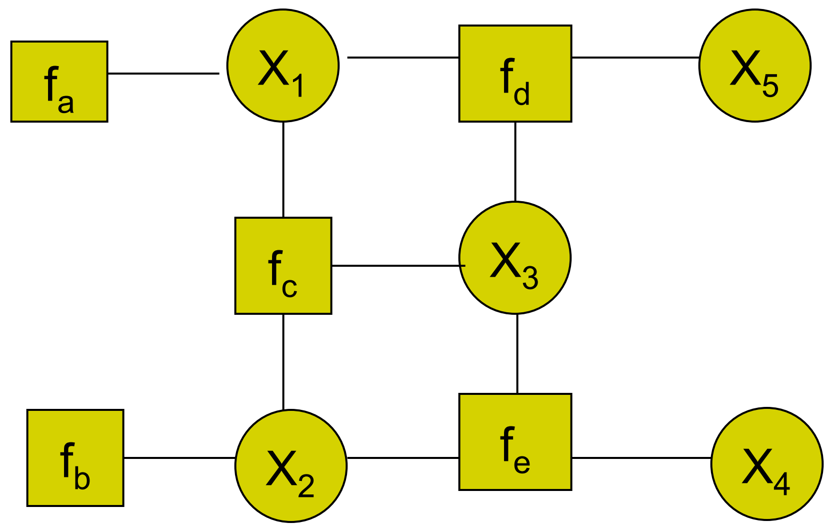
Equivalence of directed and undirected trees

- 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: $$\psi(x_r) = p(x_r); \quad \psi(x_i, x_j) = p(x_j | x_i);$$ $$Z = 1, \quad \psi(x_i) = 1$$

Factor Graph Examples

- Example 1



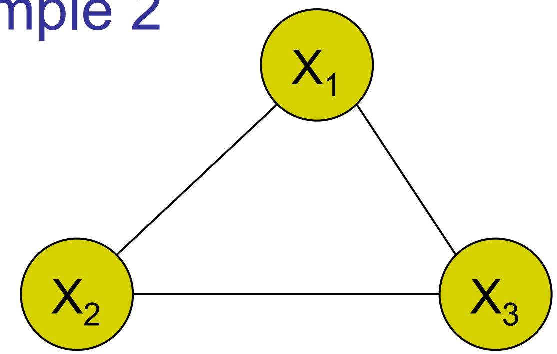


$$P(X_1) \quad P(X_2) \quad P(X_3|X_1,X_2) \quad P(X_5|X_1,X_3) \quad P(X_4|X_2,X_3)$$

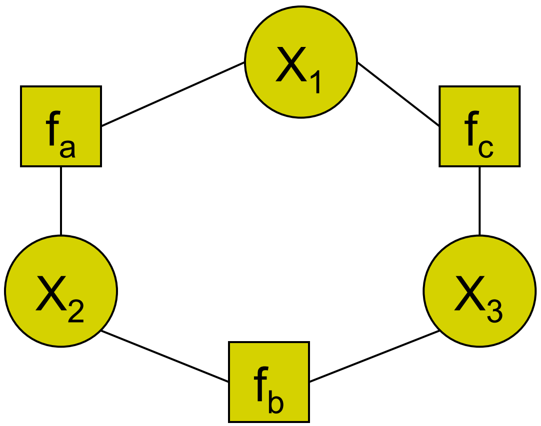
$$f_a(X_1) \quad f_b(X_2) \quad f_c(X_3,X_1,X_2) \quad f_d(X_5,X_1,X_3) \quad f_e(X_4,X_2,X_3)$$

Factor Graph Examples

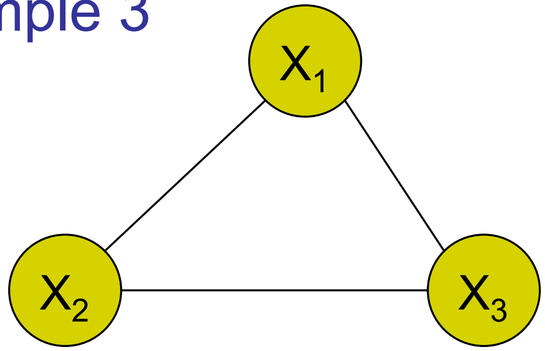
- Example 2



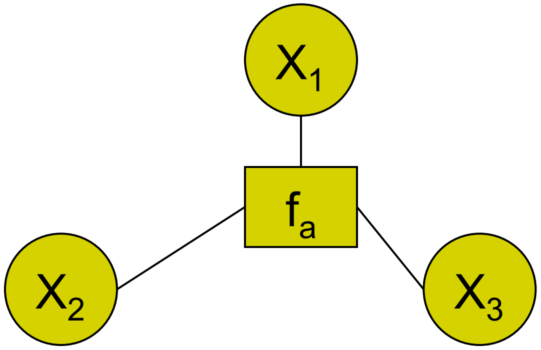
$$\psi(x_1,x_2,x_3) = f_a(x_1,x_2)f_b(x_2,x_3)f_c(x_3,x_1)$$



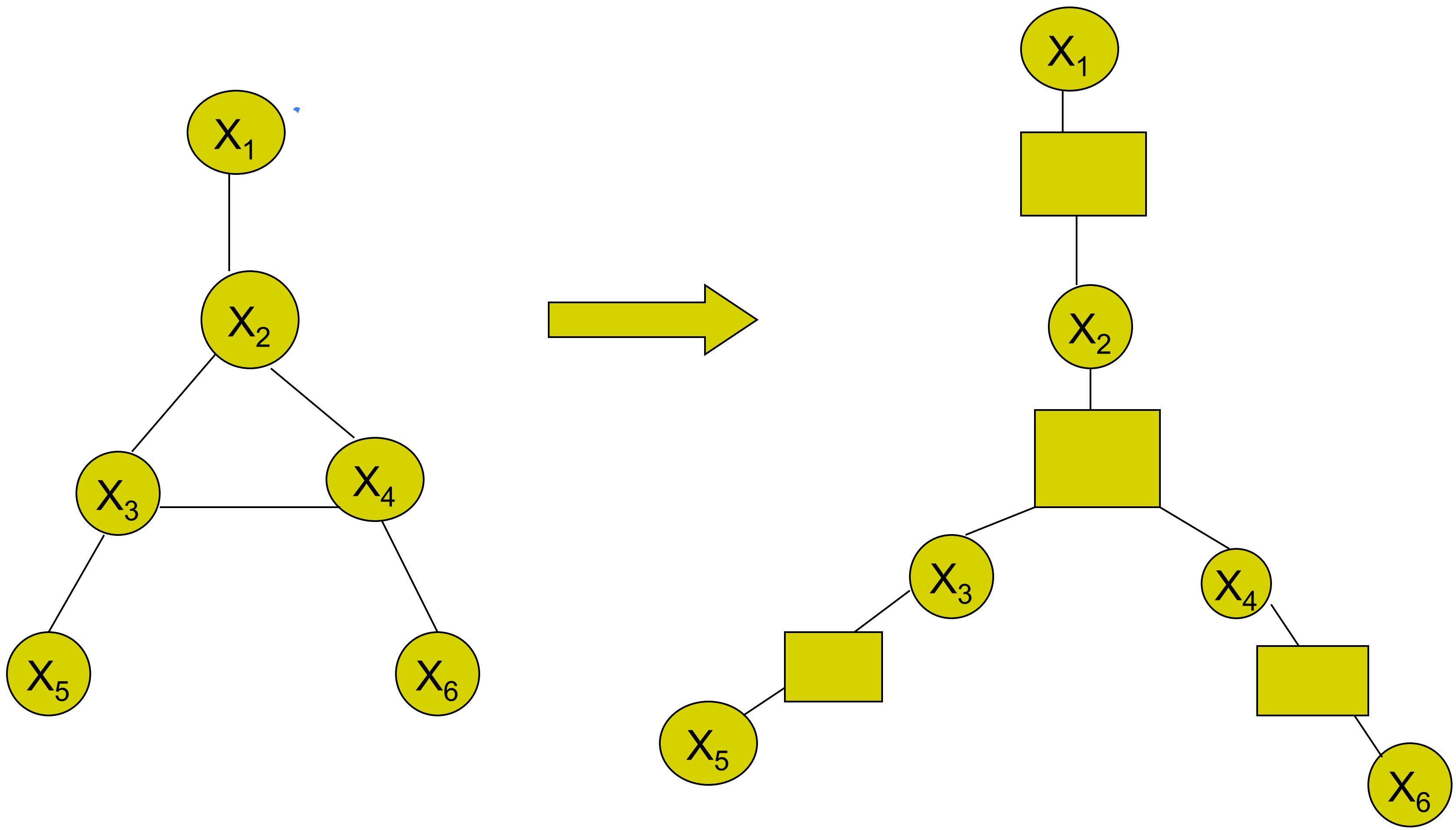
- Example 3



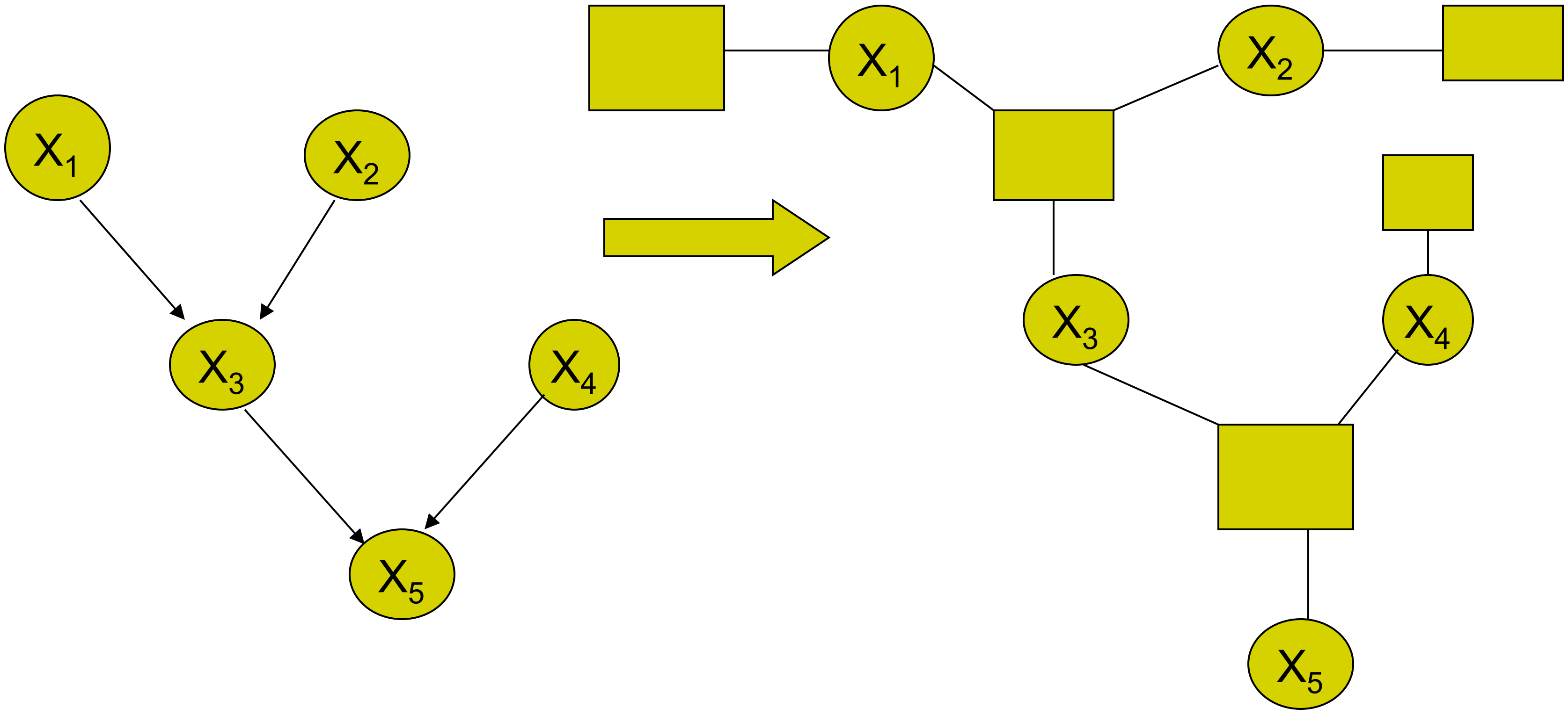
$$\psi(x_1,x_2,x_3) = f_a(x_1,x_2,x_3)$$



Tree-like Undirected GMs to Factor Trees

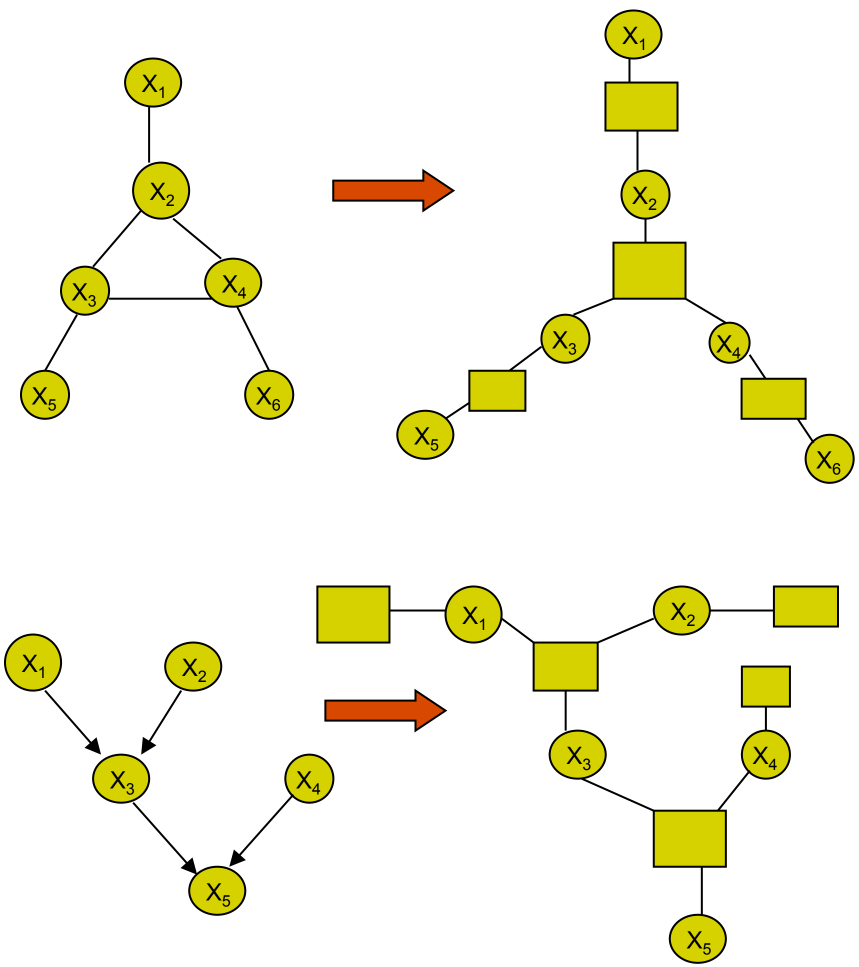


Poly-trees to Factor trees



Why factor graphs?

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



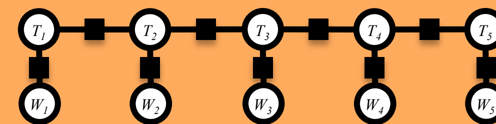
Exact Inference

1. Data

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

2. Model

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



3. Objective

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

5. Inference

1. Marginal Inference

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

2. Partition Function

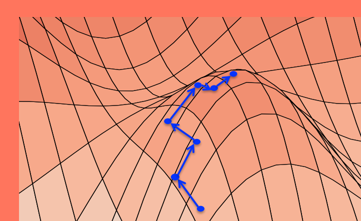
$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference

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

4. Learning

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

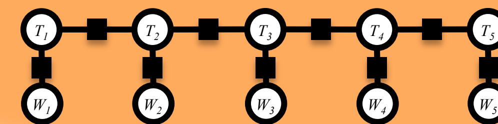


Goals for Today's Lecture

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

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



Objective

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

5. Inference

1. Marginal Inference

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

2. Partition Function

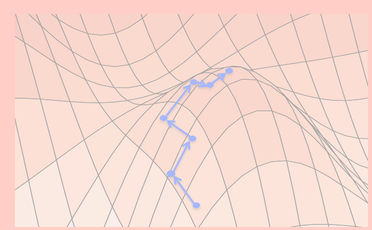
$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference

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

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 cliques

$$p(x_i) = \sum_{\boldsymbol{x}' : x'_i = x_i} p(\boldsymbol{x}' \mid \boldsymbol{\theta}) \qquad \Big| \qquad p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function
Compute the normalization constant

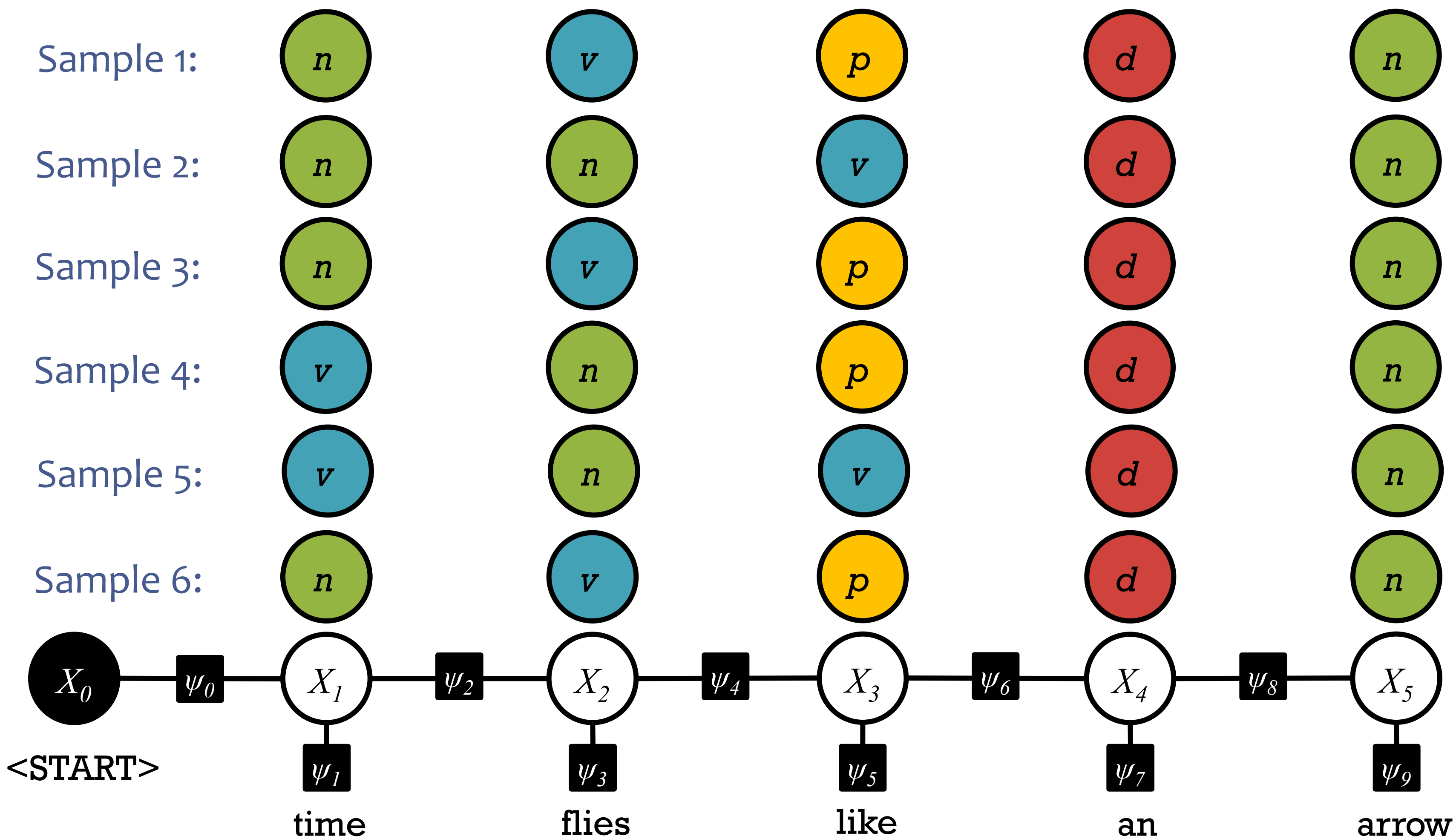
$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference
Compute variable assignment with highest probability

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\text{argmax}} \; p(\boldsymbol{x} \mid \boldsymbol{\theta})$$

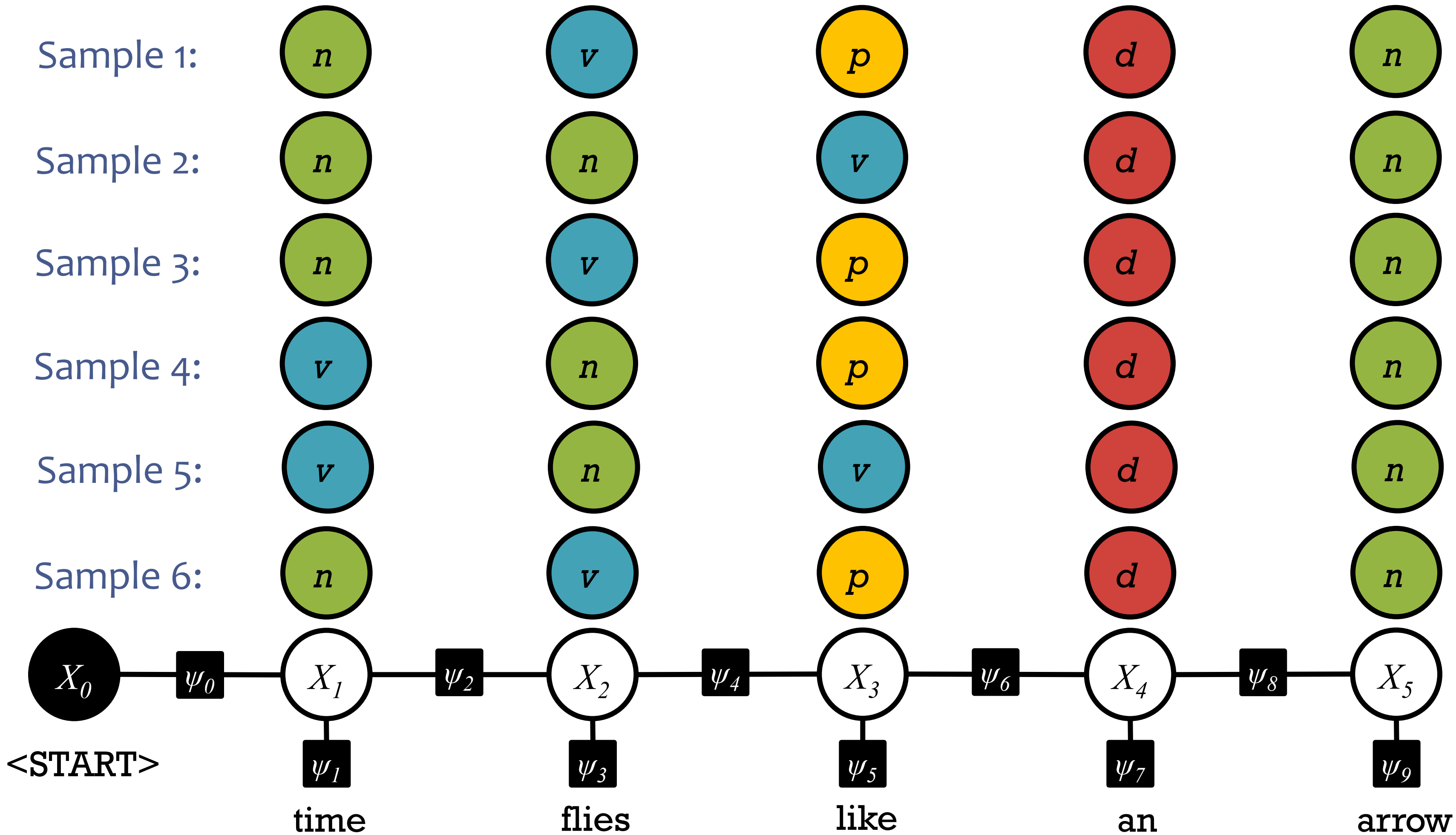
Marginals by Sampling on Factor Graph

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

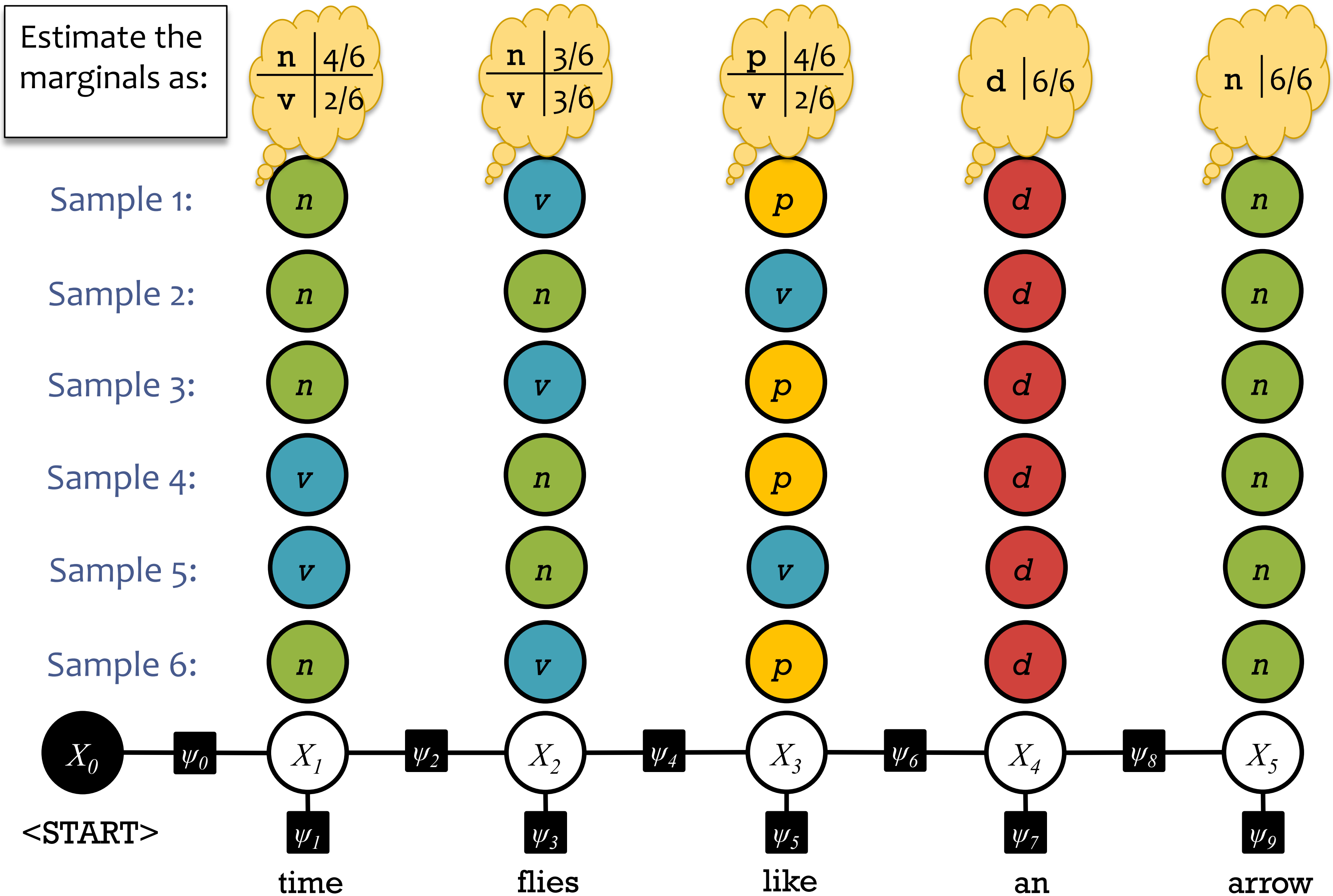


Marginals by Sampling on Factor Graph

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



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("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

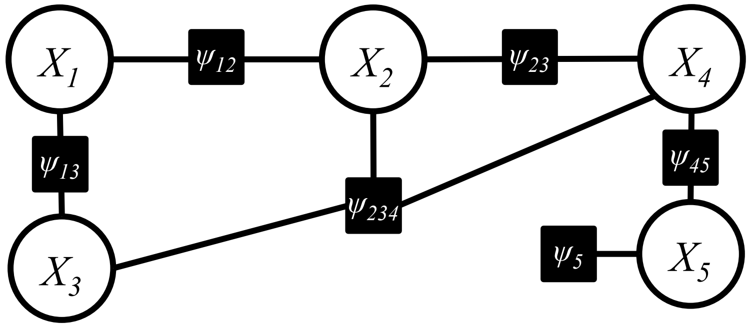
The Variable Elimination Algorithm

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:

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



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

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

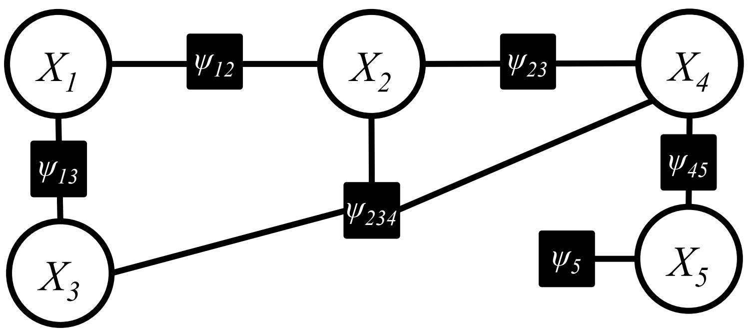
The Variable Elimination Algorithm

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:

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



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

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

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

x_1	x_2	x_3	x_4	x_5	$p(\boldsymbol{x})$
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
…	…	…	…	…	…

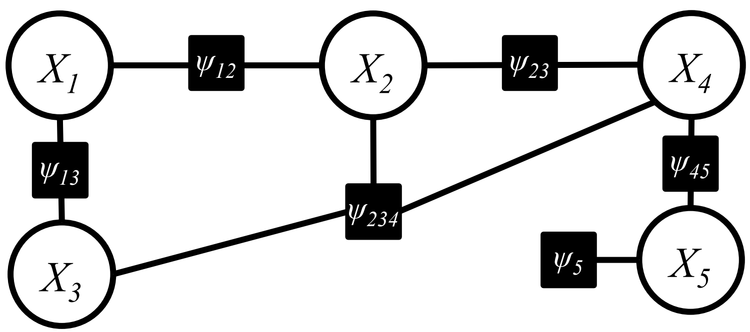
The Variable Elimination Algorithm

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:

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



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

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

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

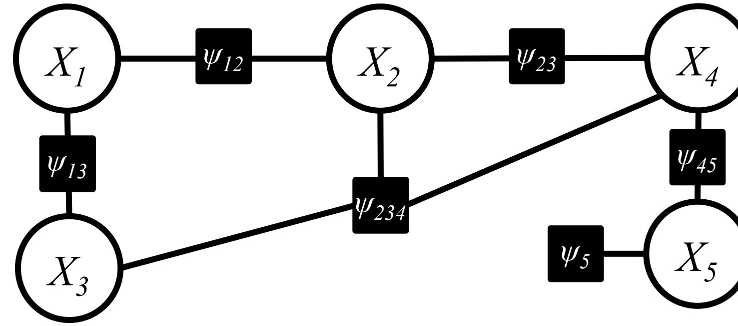
x_1	x_2	x_3	x_4	x_5	$p(\mathbf{x})$
0	0	0	0	0	0.019517693
0	0	0	0	1	0.017090249
0	0	0	0	2	0.014885825
0	0	0	1	0	0.024117638
0	0	0	1	1	0.000925849
0	0	0	1	2	0.028112576
0	0	0	2	0	0.028050205
0	0	0	2	1	0.004812689
0	0	0	2	2	0.007987737
0	0	1	0	0	0.028433687
0	0	1	0	1	0.037073469
0	0	1	0	2	0.013558227
0	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 $p(\mathbf{x})$.



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

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

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

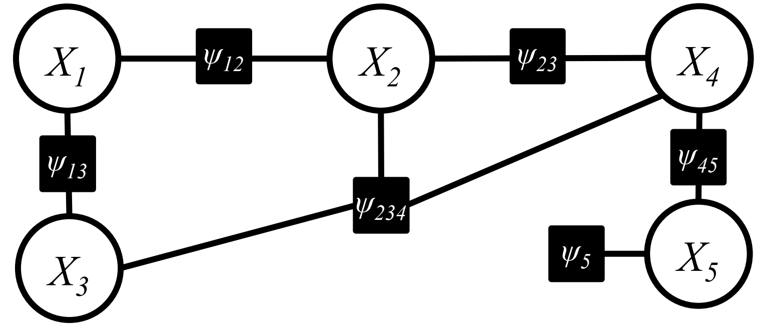
We denote the **marginal's table** by $m_5(x_4)$.

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

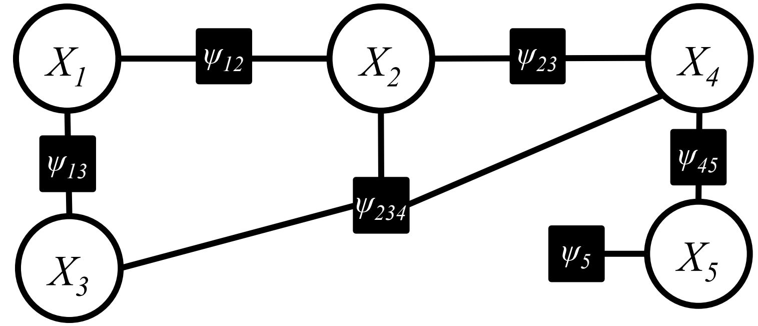
Instead, capitalize on the factorization of *p*(***x***).



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

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

The Variable Elimination Algorithm



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

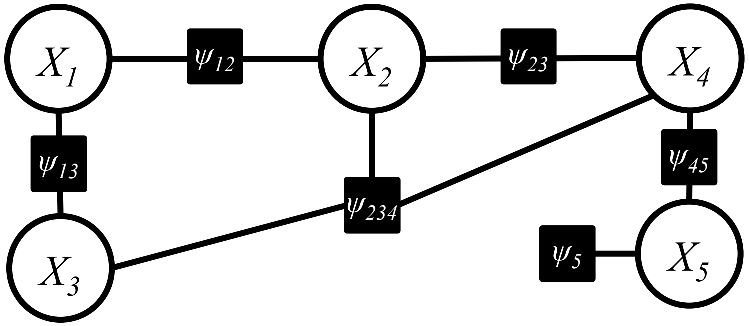
$$
\begin{aligned}
Z &= \sum_{x_1}\sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} \psi_{12}(x_1,x_2)\psi_{13}(x_1,x_3)\psi_{24}(x_2,x_4)\psi_{234}(x_2,x_3,x_4)\psi_{45}(x_4,x_5)\psi_5(x_5) \\
&= \sum_{x_1}\sum_{x_2}\sum_{x_3}\sum_{x_4} \psi_{12}(x_1,x_2)\psi_{13}(x_1,x_3)\psi_{24}(x_2,x_4)\psi_{234}(x_2,x_3,x_4)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

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

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

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

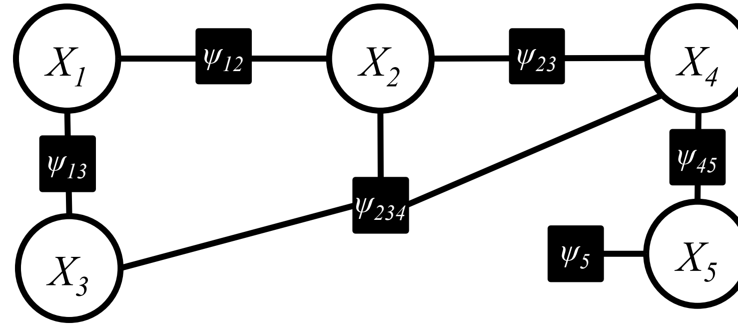
3 additions

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

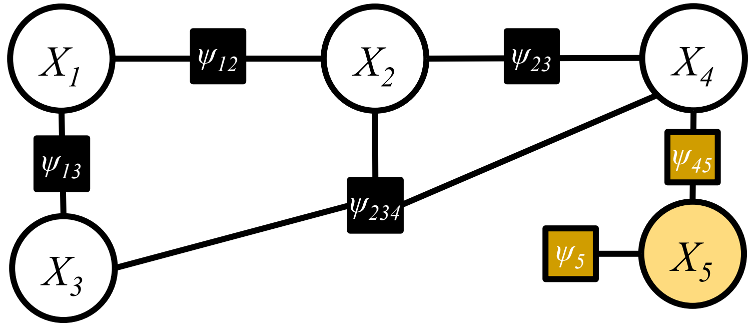
3^2 additions

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



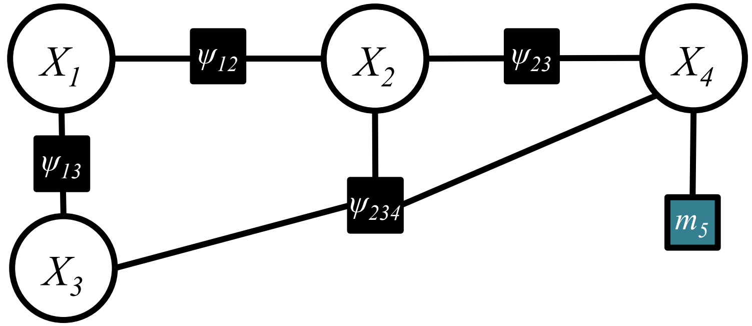
$$Z = \sum_{x_1}\sum_{x_2}\psi_{12}(x_1,x_2)\sum_{x_3}\psi_{13}(x_1,x_3)\sum_{x_4}\psi_{24}(x_2,x_4)\psi_{234}(x_2,x_3,x_4)\sum_{x_5}\psi_{45}(x_4,x_5)\psi_5(x_5)$$

$$= \sum_{x_1}\sum_{x_2}\psi_{12}(x_1,x_2)\sum_{x_3}\psi_{13}(x_1,x_3)\sum_{x_4}\psi_{24}(x_2,x_4)\psi_{234}(x_2,x_3,x_4)m_5(x_4)$$

$$= \sum_{x_1}\sum_{x_2}\psi_{12}(x_1,x_2)\sum_{x_3}\psi_{13}(x_1,x_3)m_4(x_2,x_3)$$

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

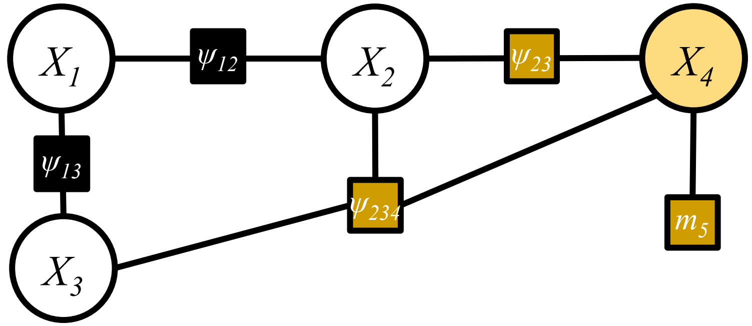
The Variable Elimination Algorithm



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

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

The Variable Elimination Algorithm



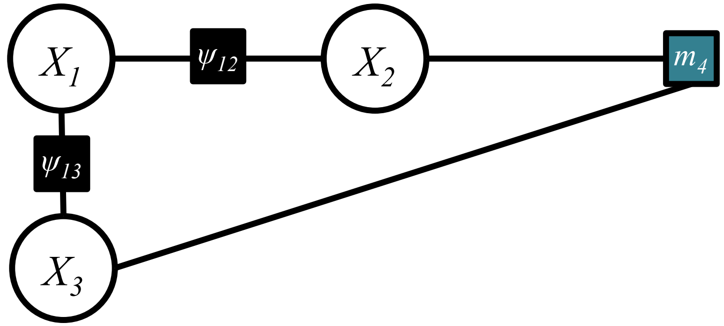
$$Z = \sum_{x_1}\sum_{x_2}\psi_{12}(x_1,x_2)\sum_{x_3}\psi_{13}(x_1,x_3)\sum_{x_4}\psi_{24}(x_2,x_4)\psi_{234}(x_2,x_3,x_4)\sum_{x_5}\psi_{45}(x_4,x_5)\psi_5(x_5)$$

$$= \sum_{x_1}\sum_{x_2}\psi_{12}(x_1,x_2)\sum_{x_3}\psi_{13}(x_1,x_3)\sum_{x_4}\psi_{24}(x_2,x_4)\psi_{234}(x_2,x_3,x_4)m_5(x_4)$$

$$= \sum_{x_1}\sum_{x_2}\psi_{12}(x_1,x_2)\sum_{x_3}\psi_{13}(x_1,x_3)m_4(x_2,x_3)$$

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

The Variable Elimination Algorithm



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

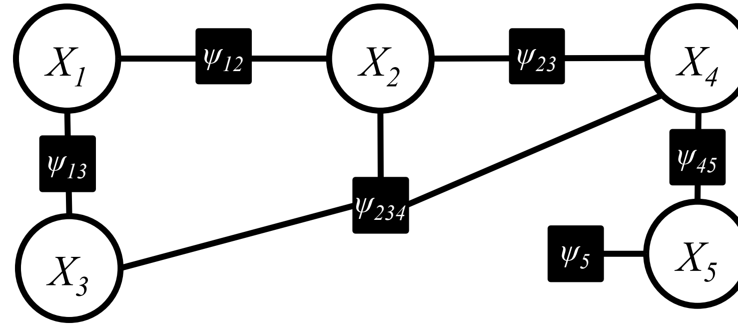
$$= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \psi_{234}(x_2, x_3, x_4) m_5(x_4)$$

$$= \sum_{x_1} \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) m_4(x_2, x_3)$$

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

Variable Elimination Complexity

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



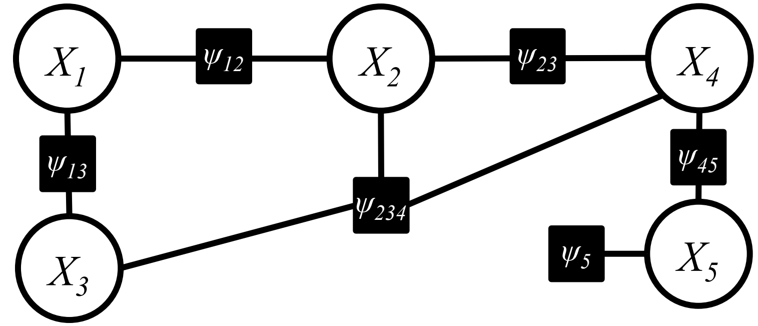
Naïve solution is $O(k^n)$

Variable elimination is $O(nk^r)$

where
n = # of variables
k = max # values a variable can take
r = # variables participating in largest "intermediate" table

Variable Elimination Complexity

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



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

Variable Elimination Complexity

- The graph elimination process we saw has an analogous interpretation on undirected graphical models, where instead of adding a few factor we add a new clique.
- The overall complexity is determined by the number of the largest elimination clique.

  - What is the largest elimination clique? – a pure graph theoretic question
  - **Tree-width** *t*: one less than the smallest achievable value of the cardinality of the largest elimination clique, ranging over all possible elimination ordering
  - “good” elimination orderings lead to **small cliques** and hence reduce complexity (what will happen if we eliminate "e" first in the above graph?)
  - Find the best elimination ordering of a graph --- NP-hard

  → Inference is NP-hard

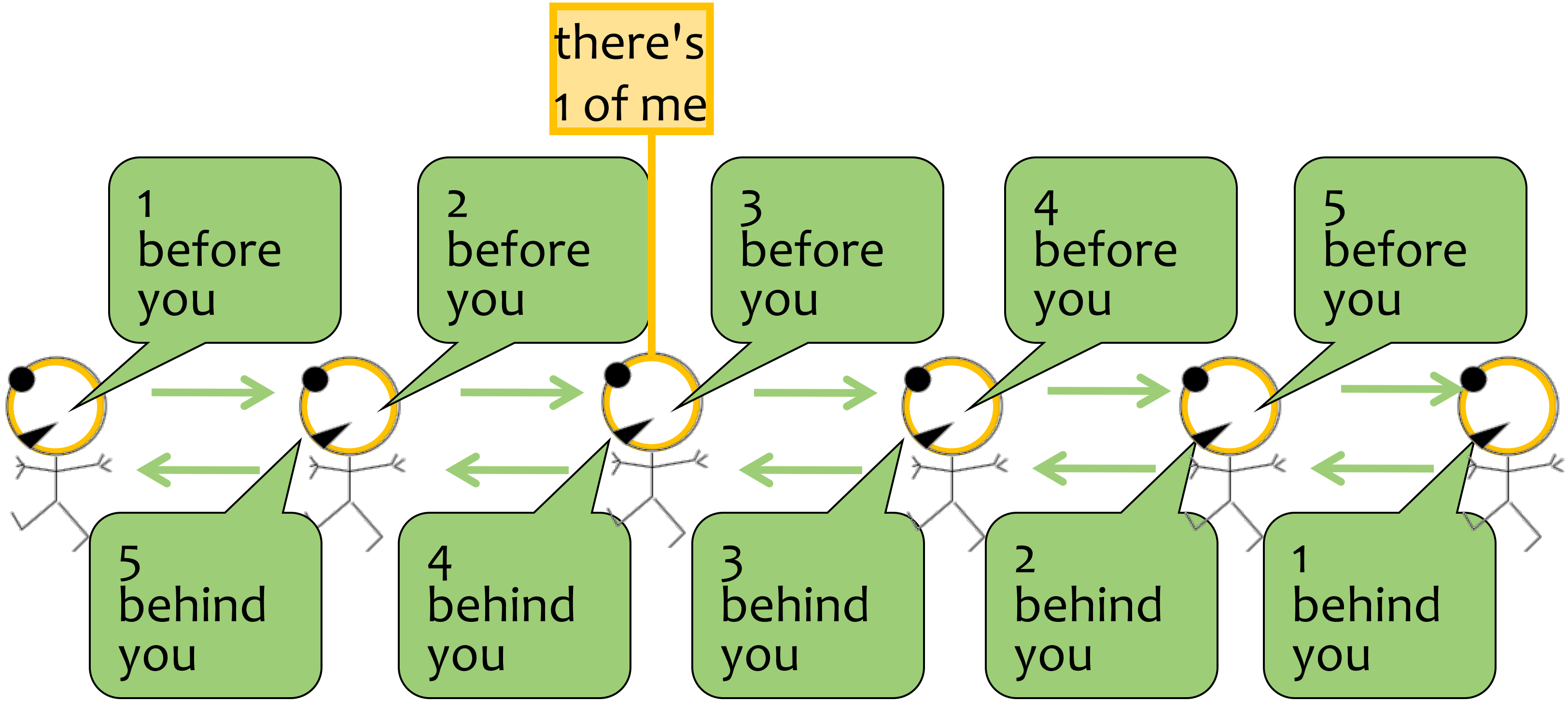
  - But there often exist "obvious" optimal or near-opt elimination ordering

Exact marginal inference for factor trees

SUM-PRODUCT BELIEF PROPAGATION

Great Ideas in ML: Message Passing

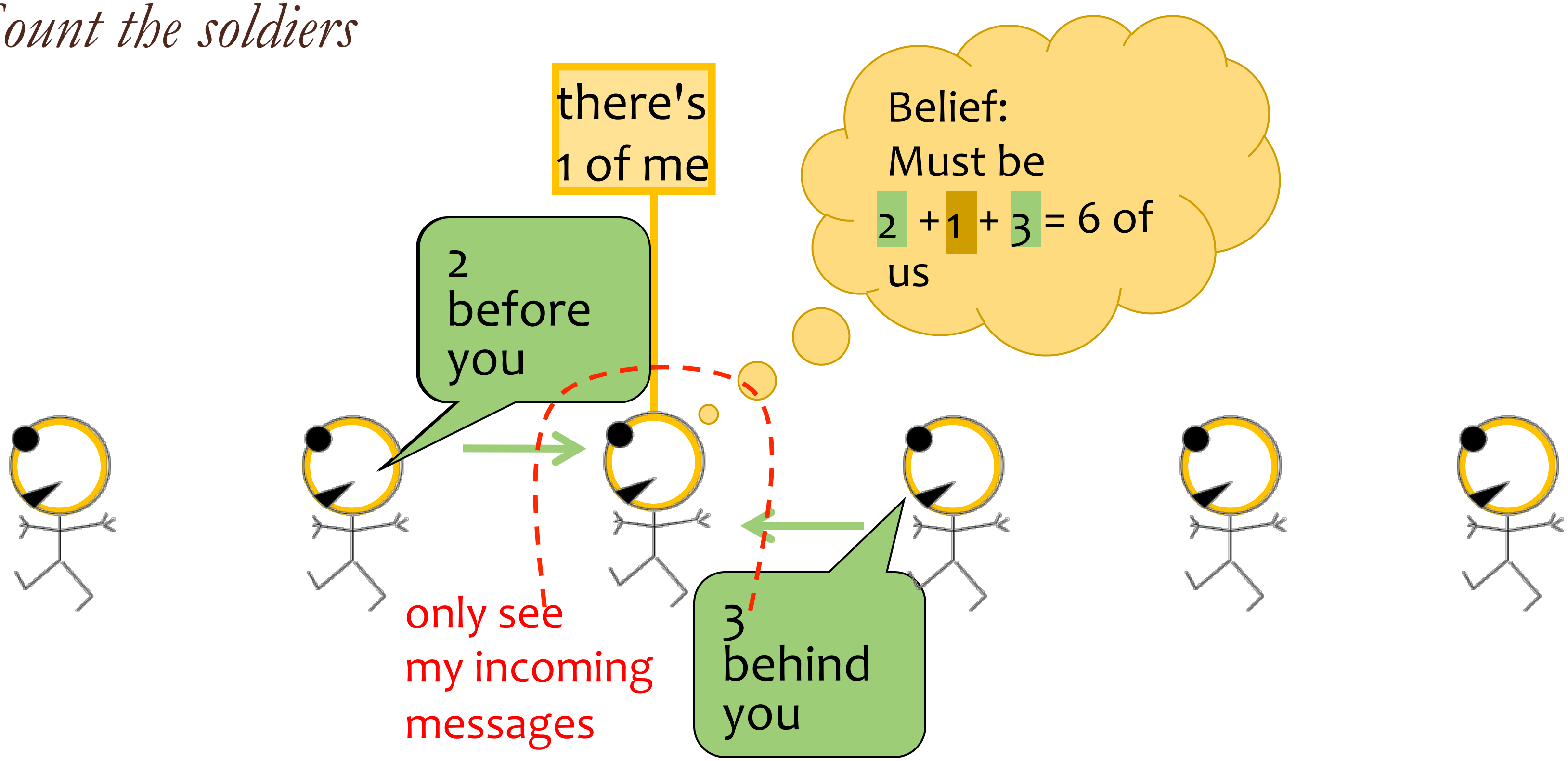
Count the soldiers



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

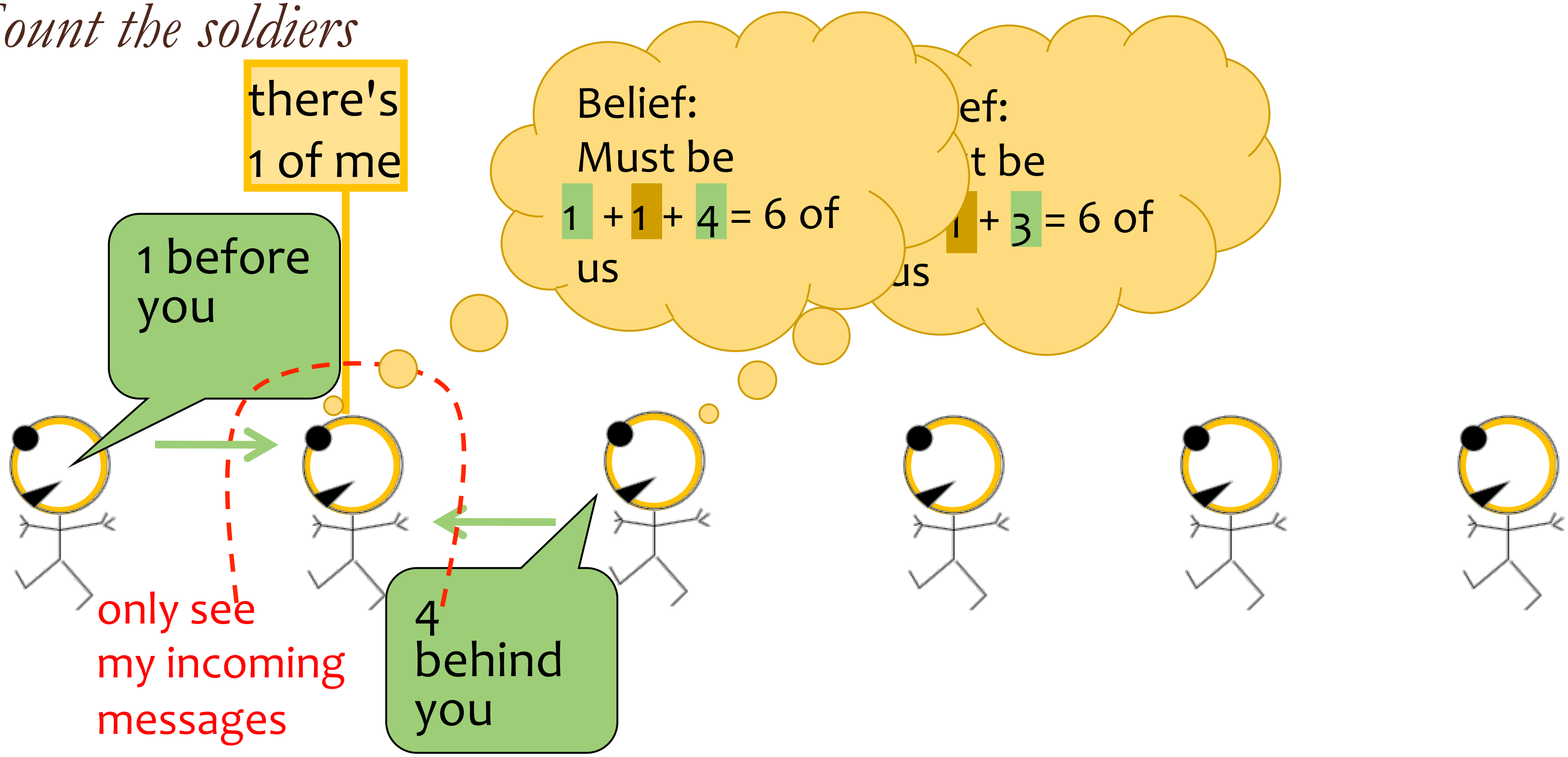
Count the soldiers



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

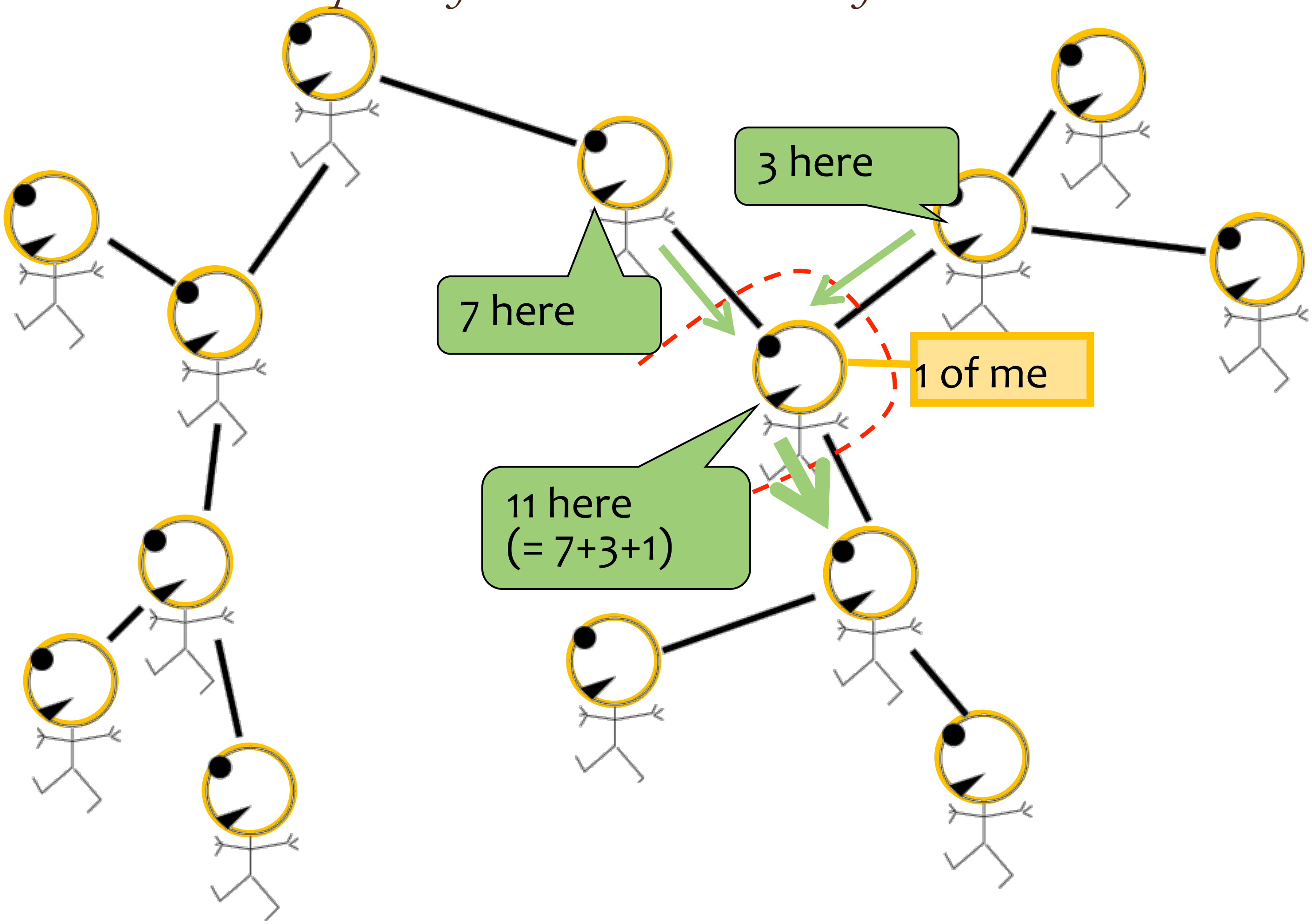
Count the soldiers



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

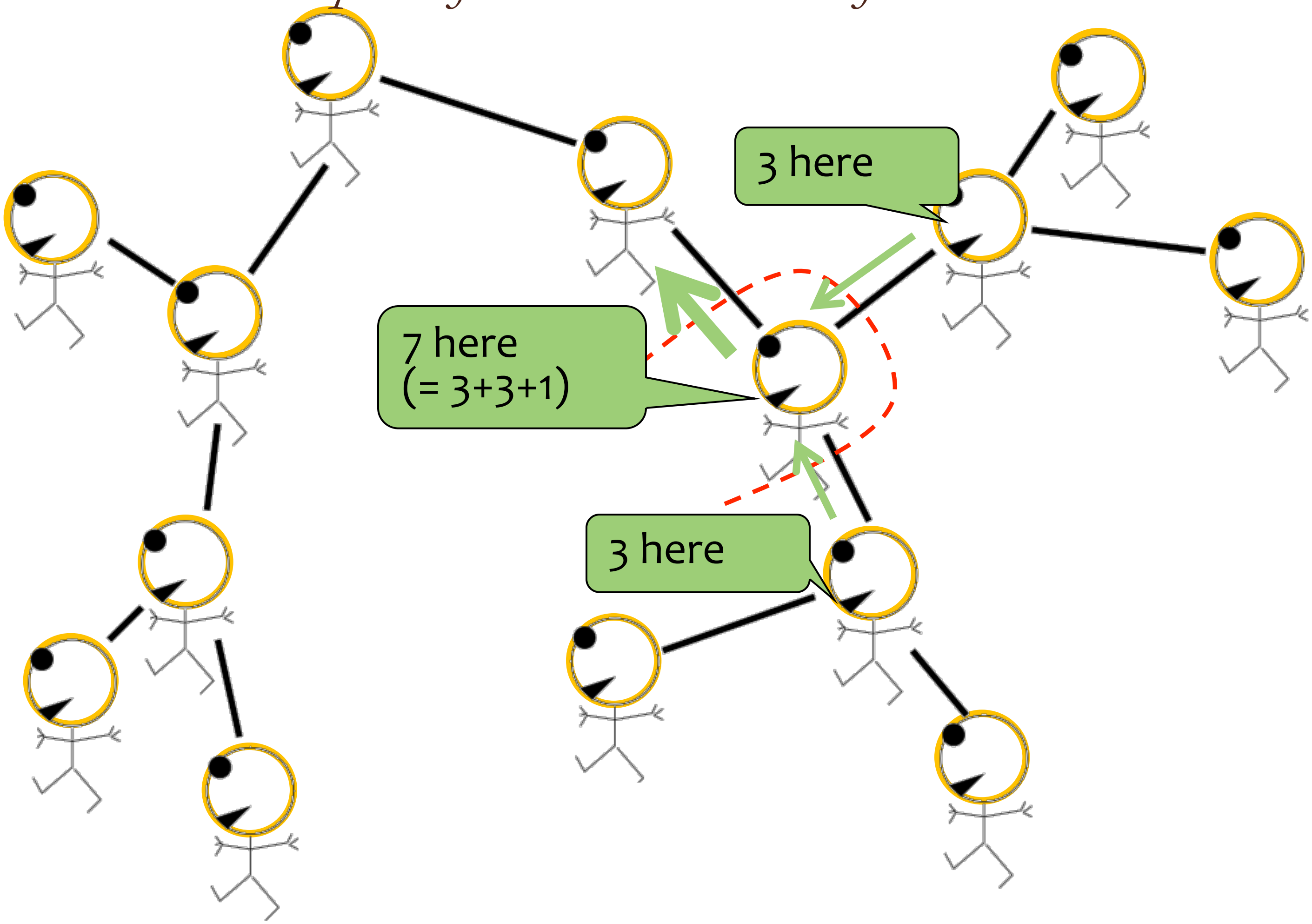
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

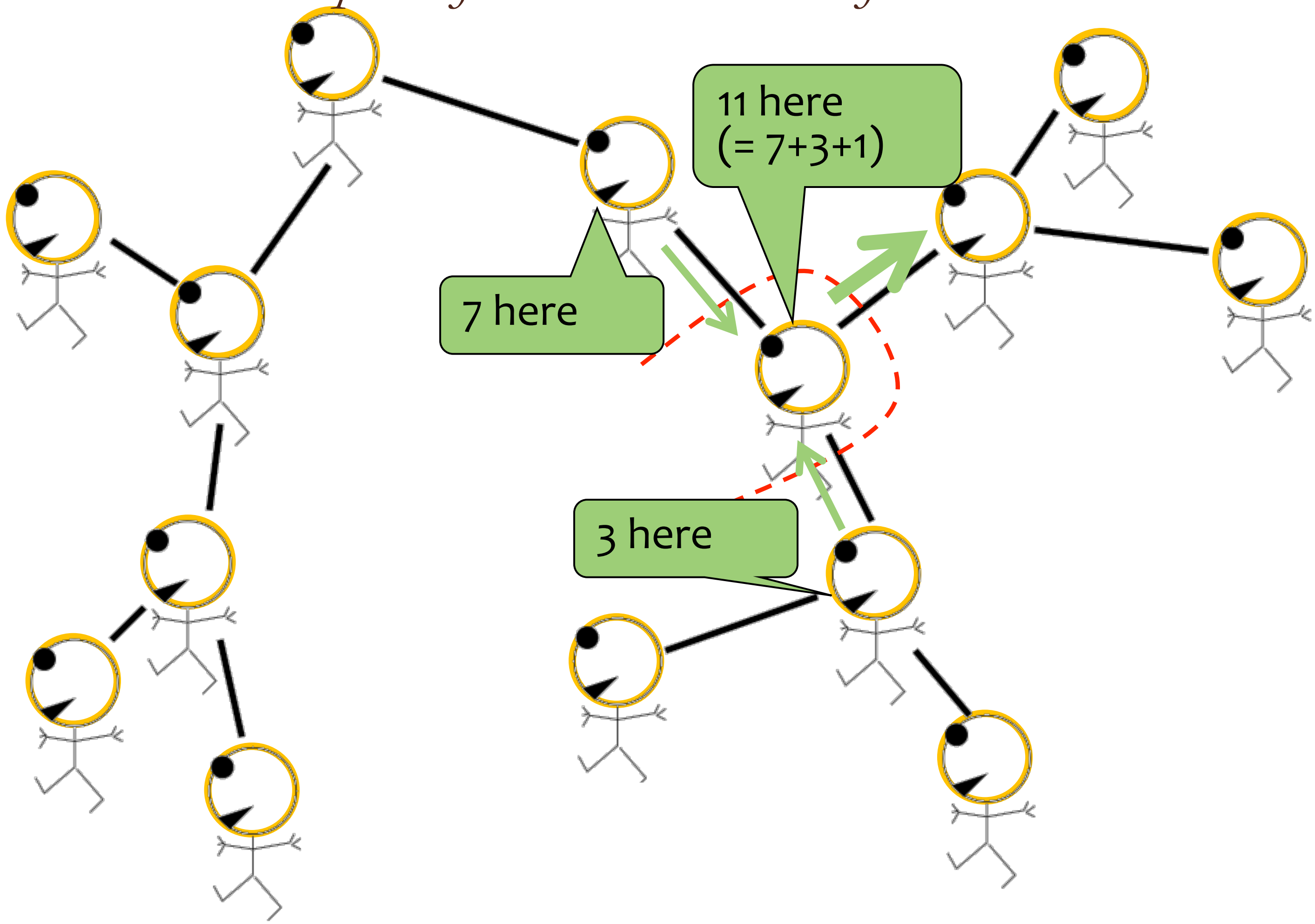
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

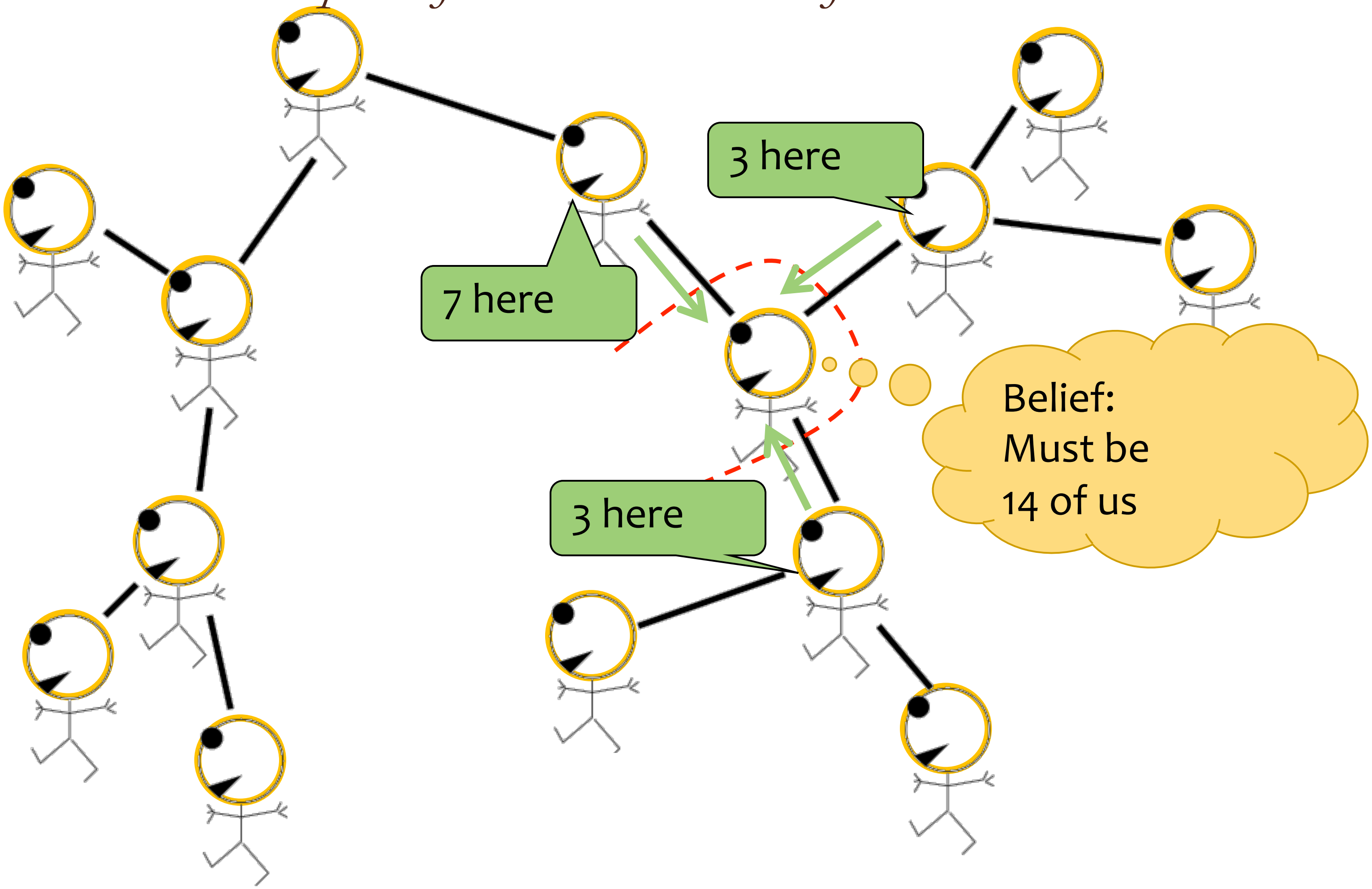
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

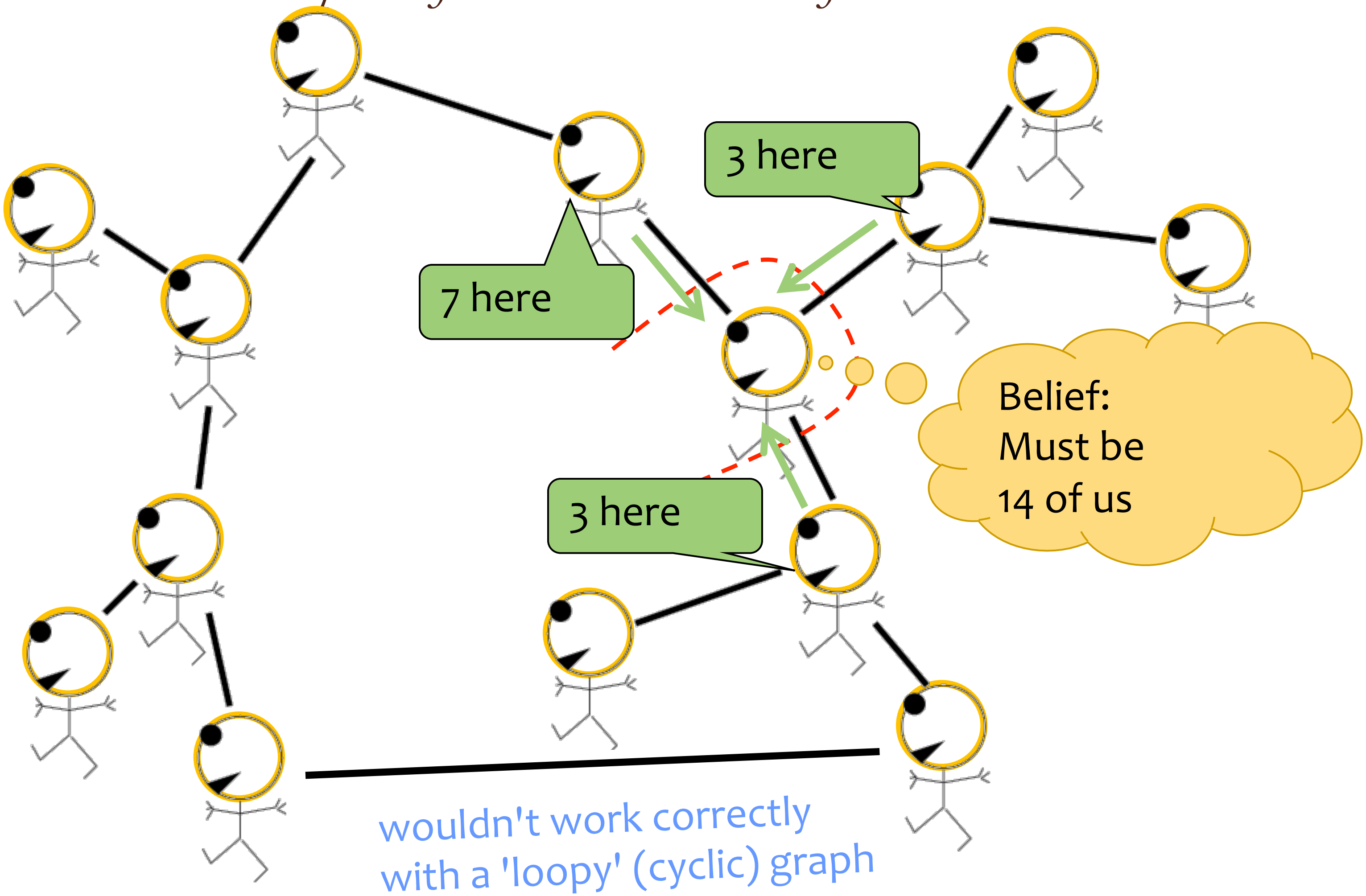
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

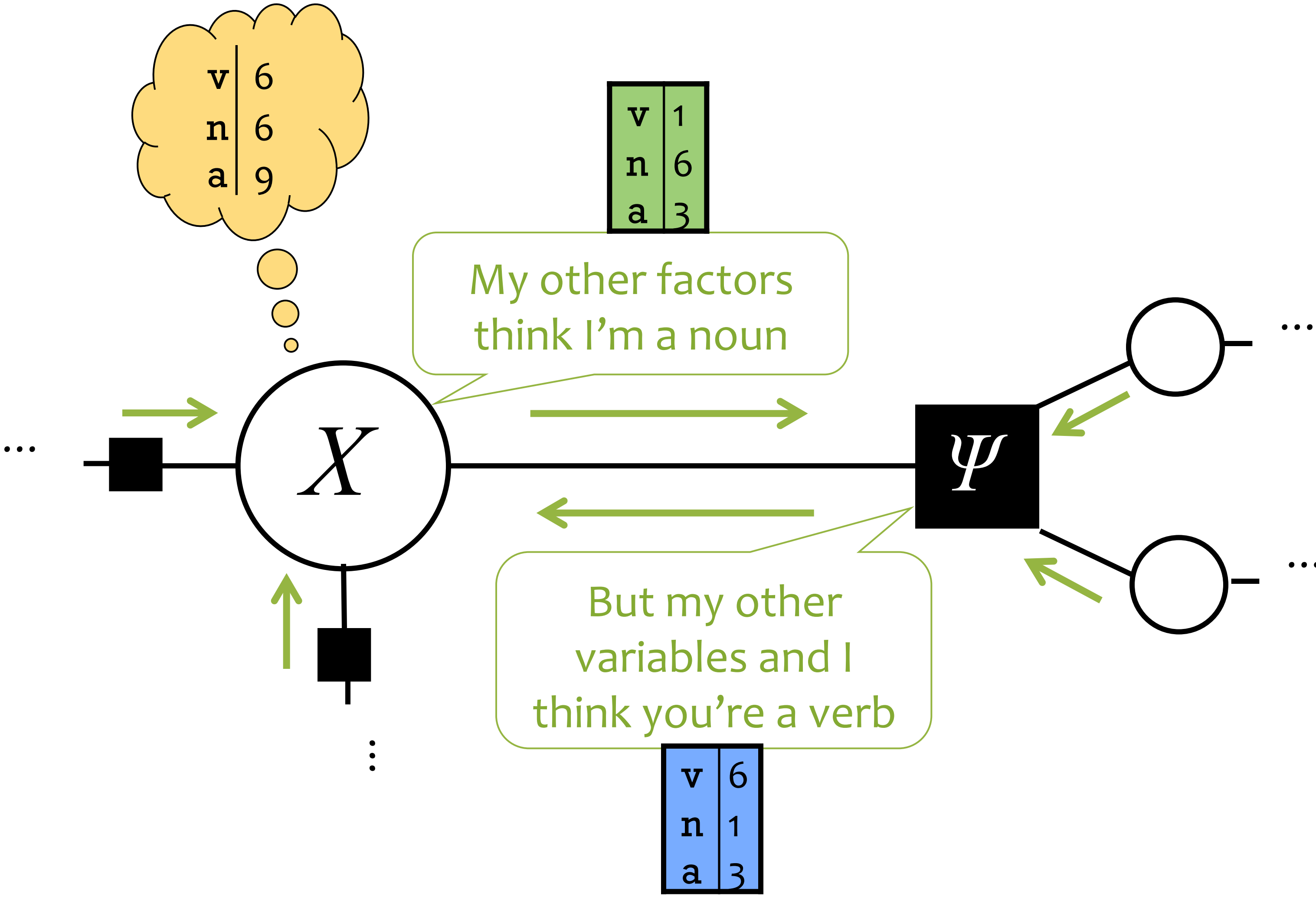
Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree



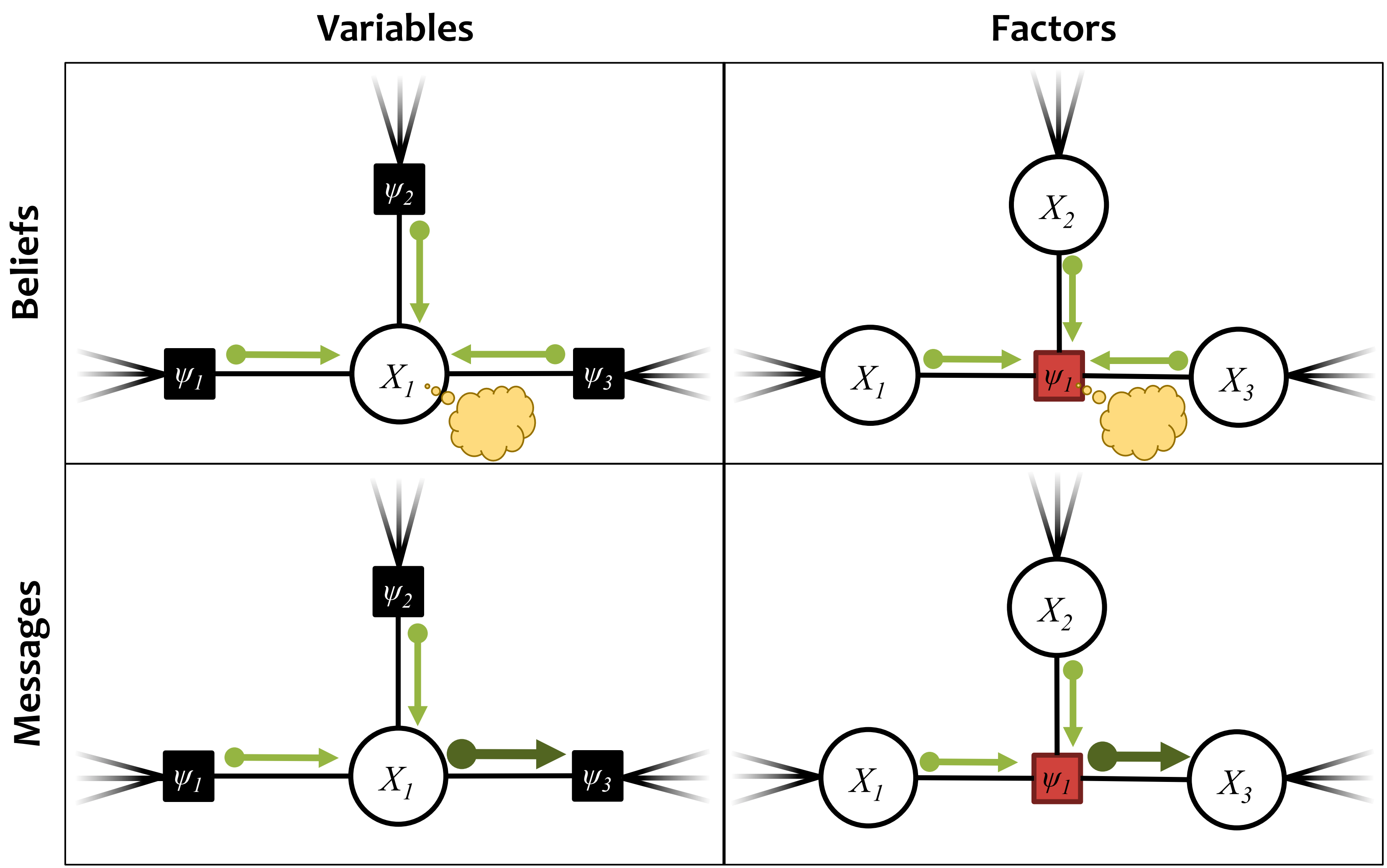
adapted from MacKay (2003) textbook

Message Passing in Belief Propagation



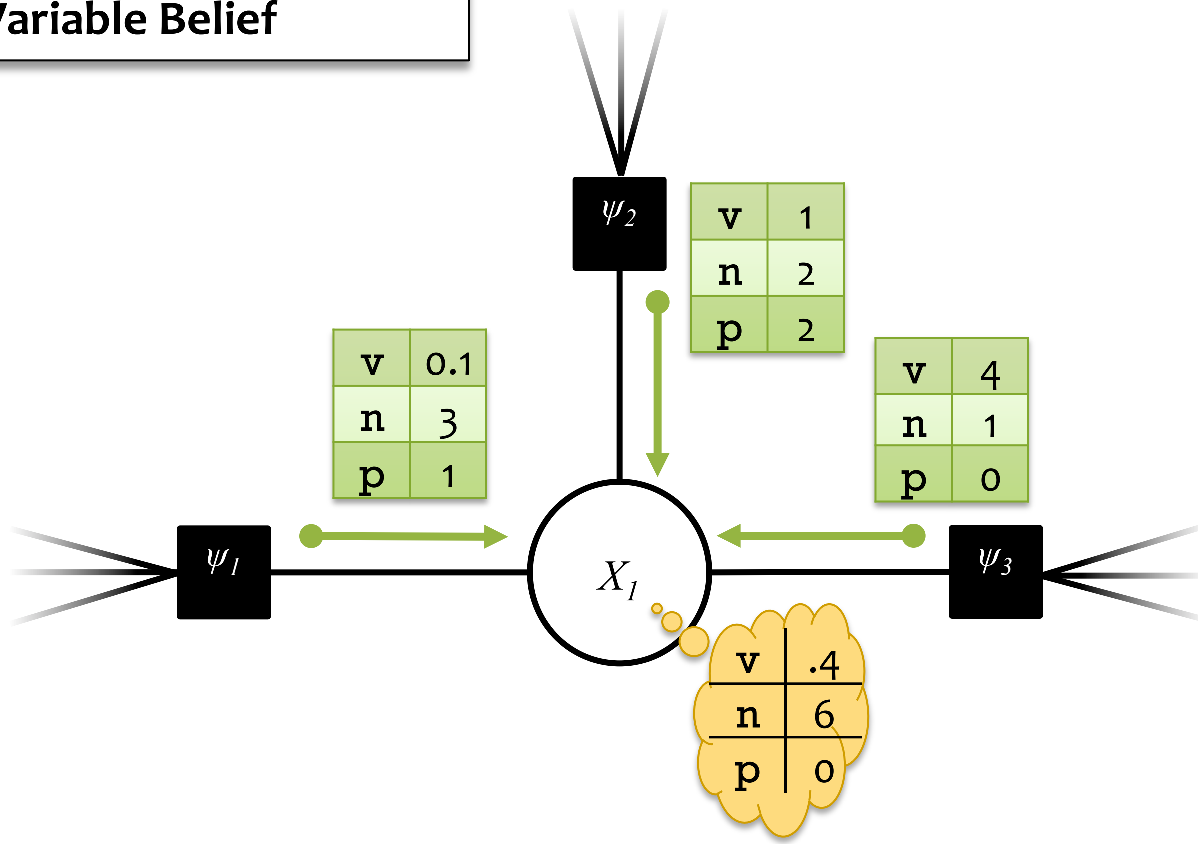
Both of these messages judge the possible values of variable X.
Their product = belief at X = product of all 3 messages to X.

Sum-Product Belief Propagation



Sum-Product Belief Propagation

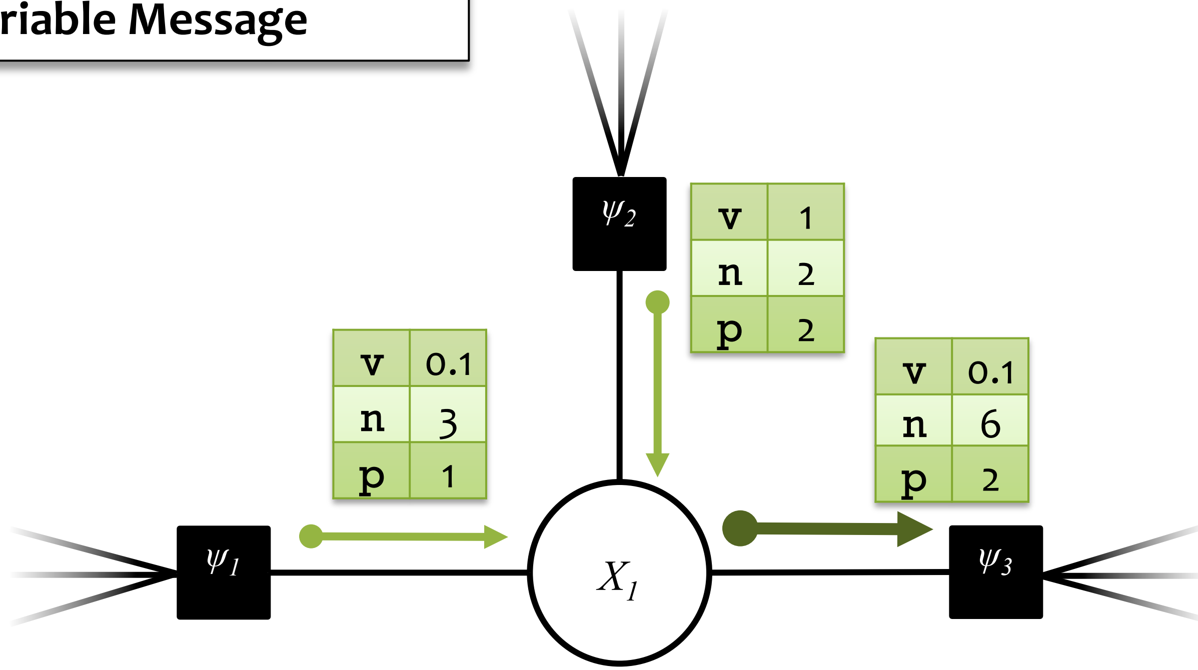
Variable Belief



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

Sum-Product Belief Propagation

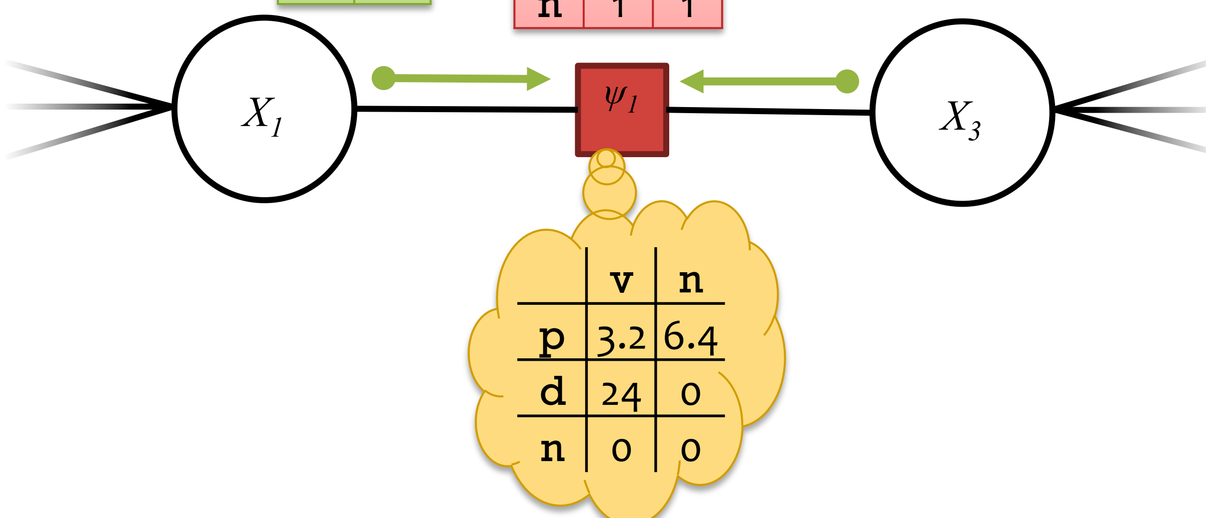
Variable Message



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

Sum-Product Belief Propagation

Factor Belief



p	4
d	1
n	0

	v	n
p	0.1	8
d	3	0
n	1	1

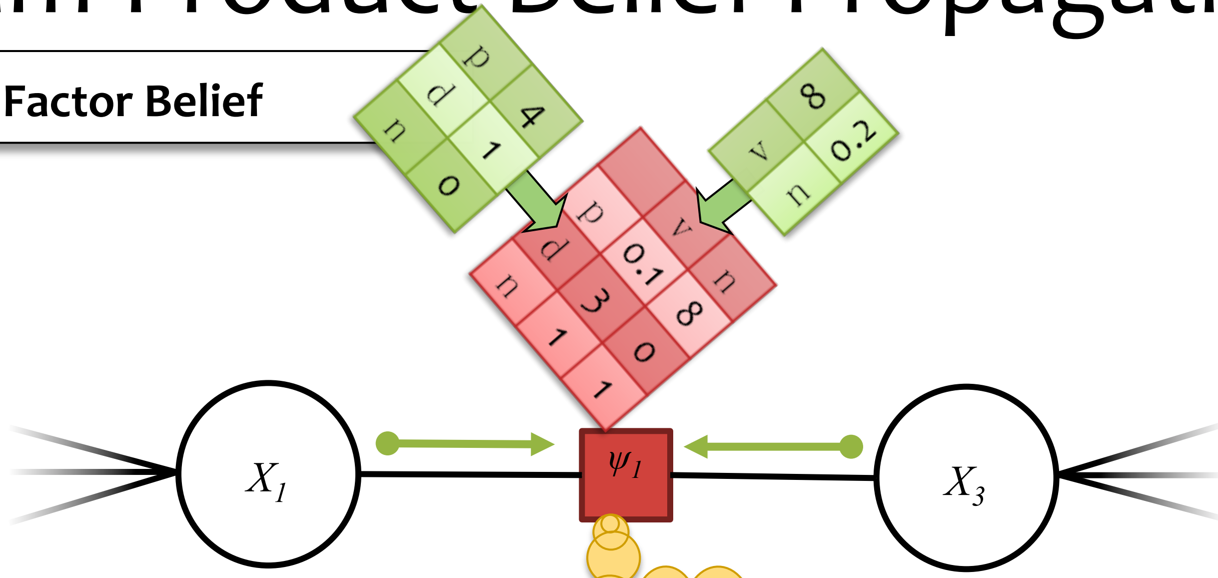
v	8
n	0.2

	v	n
p	3.2	6.4
d	24	0
n	0	0

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

Sum-Product Belief Propagation

Factor Belief



	v	n
p	3.2	6.4
d	24	0
n	0	0

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

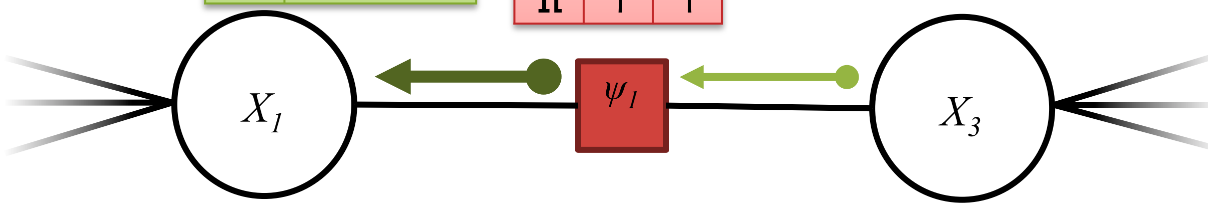
Sum-Product Belief Propagation

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

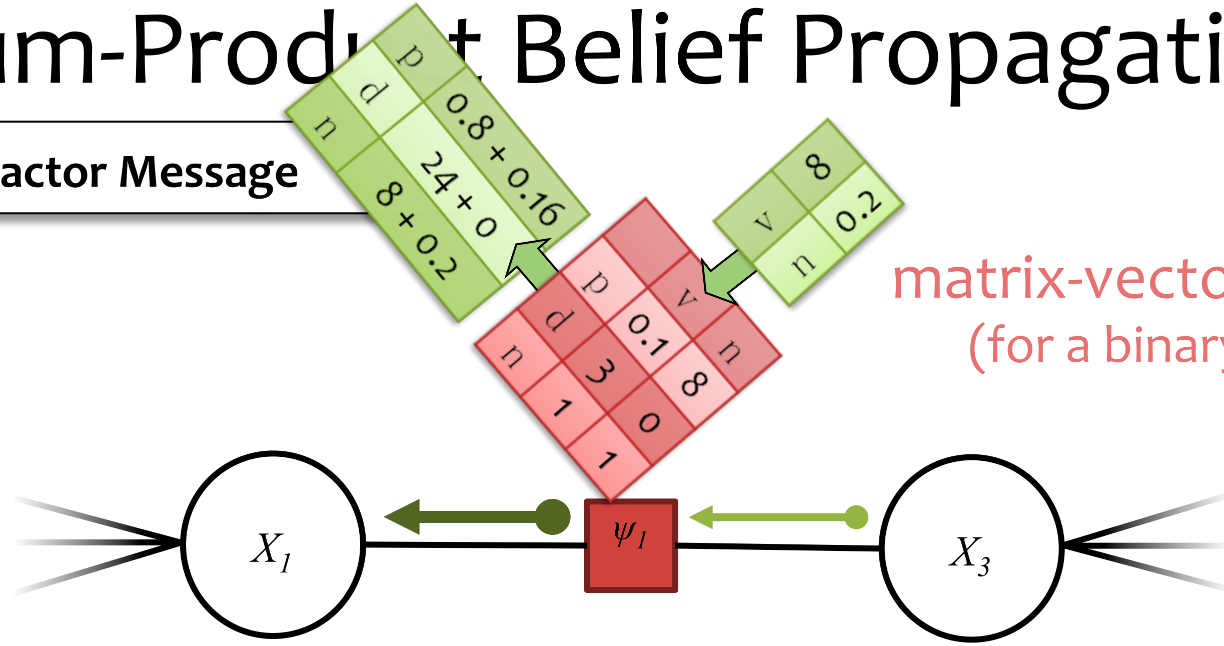
v	8
n	0.2



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

Sum-Product Belief Propagation

Factor Message



matrix-vector product
(for a binary factor)

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

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.

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

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

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

4. Compute the beliefs (unnormalized marginals).

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

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

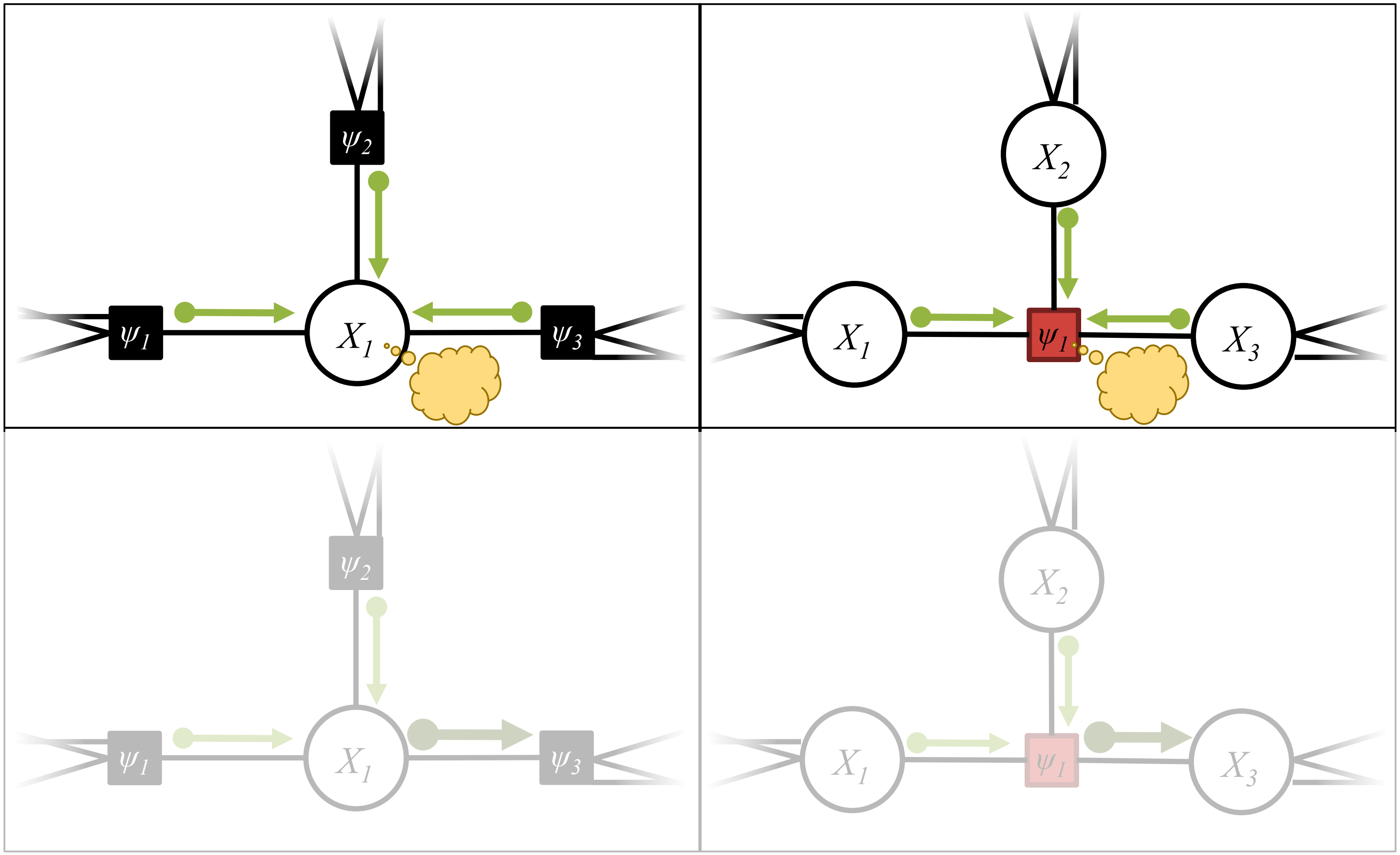
$$\boxed{p_i(x_i) \propto b_i(x_i)} \quad \boxed{p_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) \propto b_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}})}$$

Sum-Product Belief Propagation

Variables | **Factors**

Beliefs

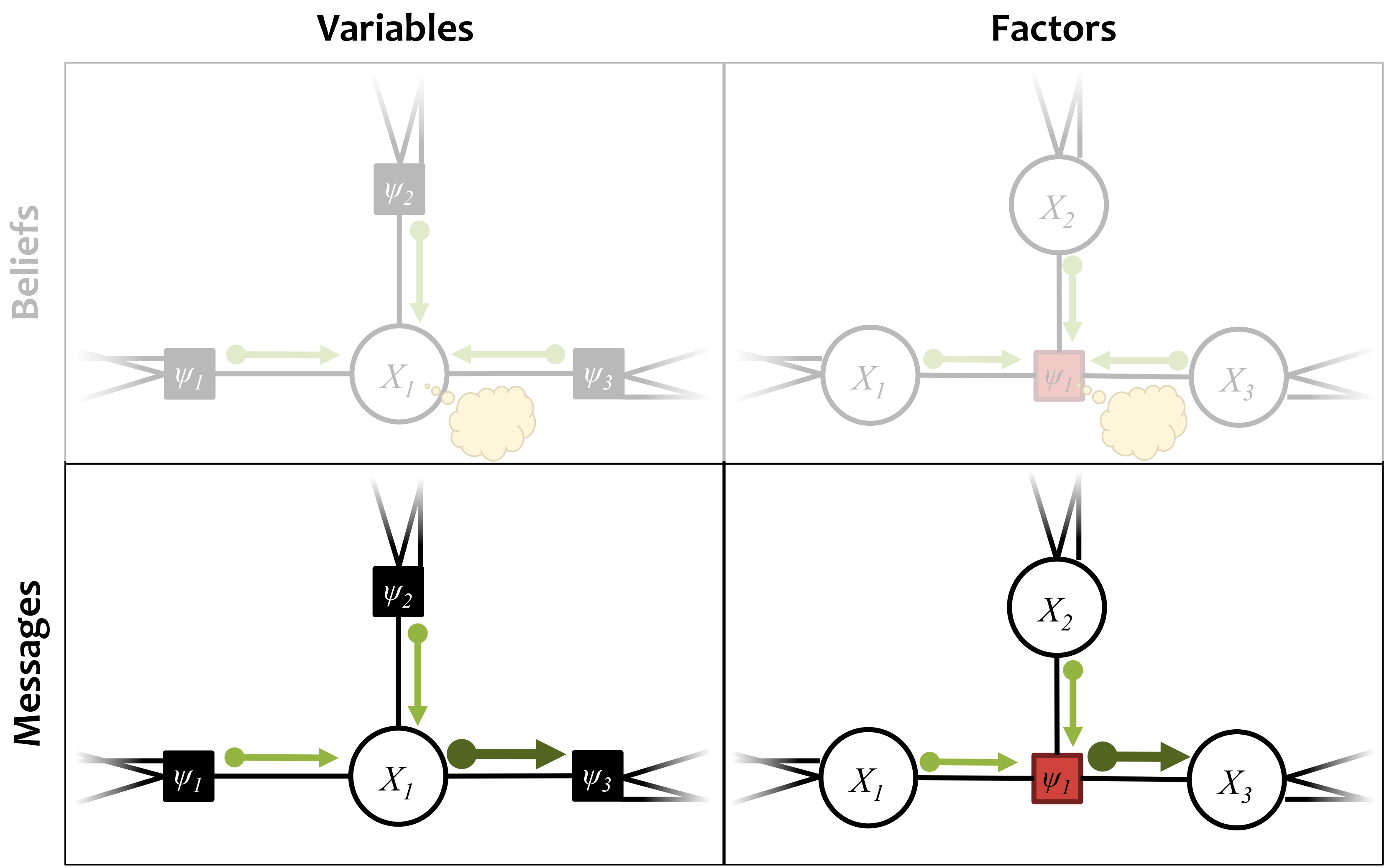
Messages



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

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

Sum-Product Belief Propagation



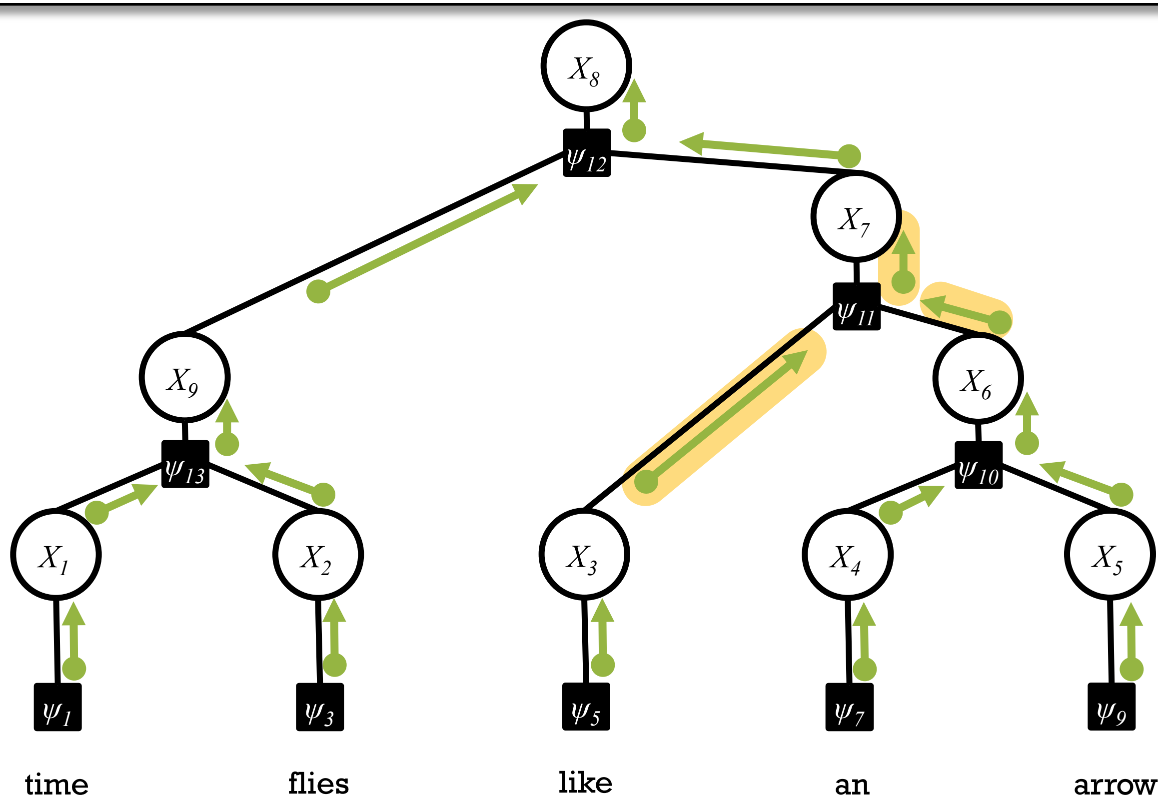
$$\mu_{i\to\alpha}(x_i) = \prod_{\alpha\in\mathcal{N}(i)\backslash\alpha} \mu_{\alpha\to i}(x_i)$$

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

(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**.

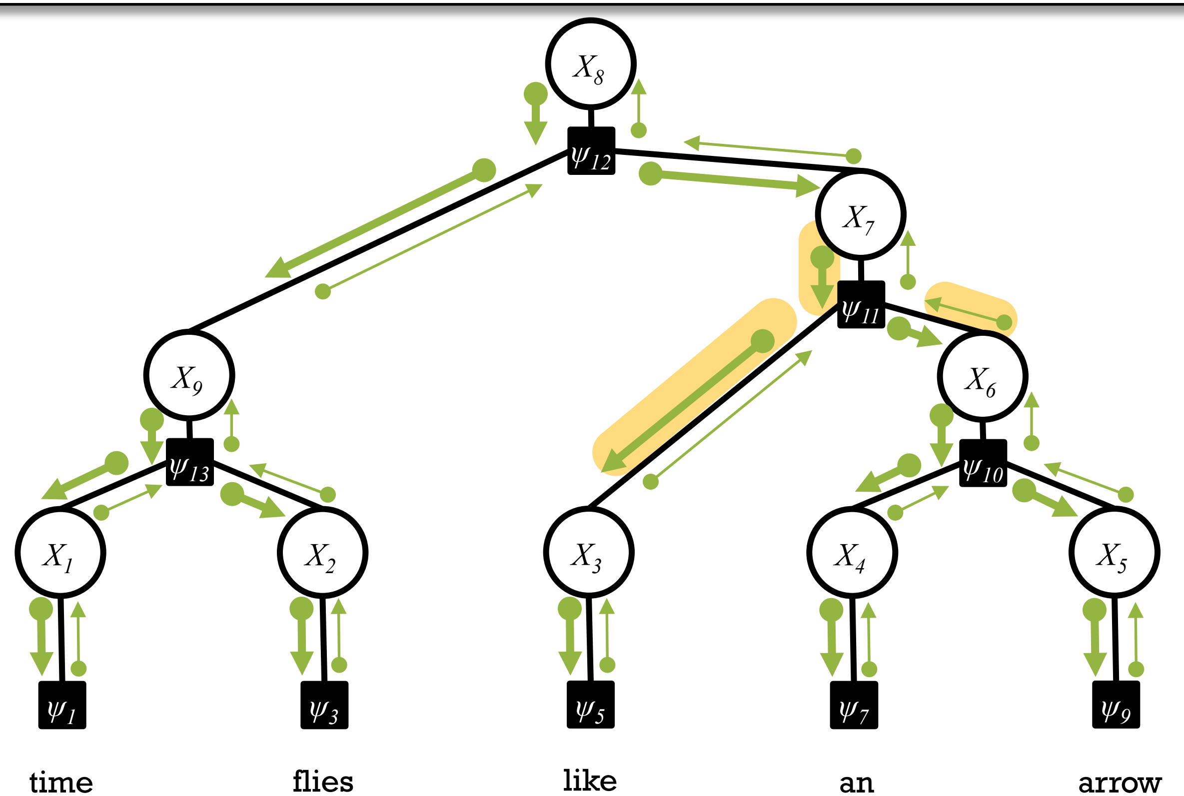
A node computes an outgoing message along an edge only after it has received incoming messages along all its other edges.



(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**.

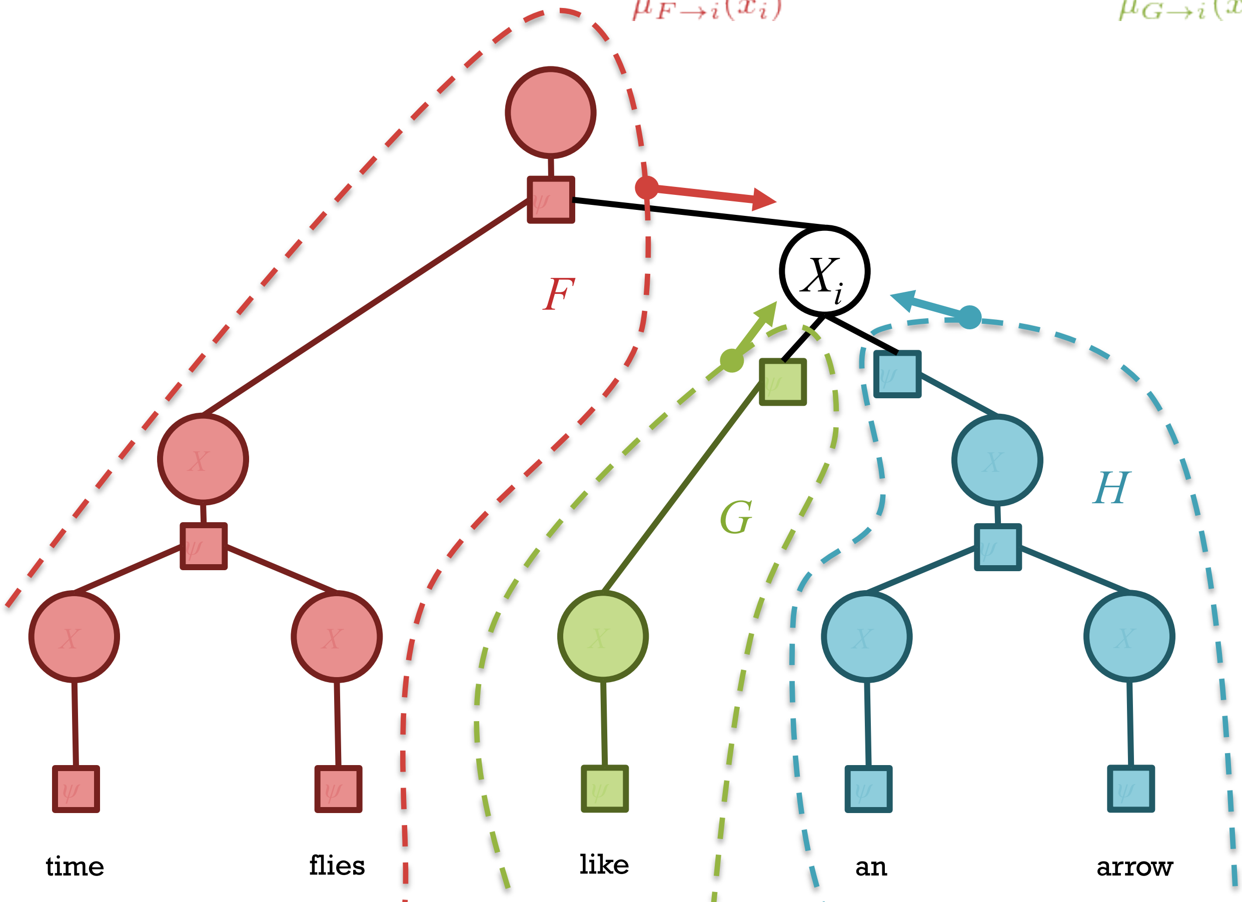
A node computes an outgoing message along an edge
only after it has received incoming messages along all its other edges.



Acyclic BP as Dynamic Programming

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

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



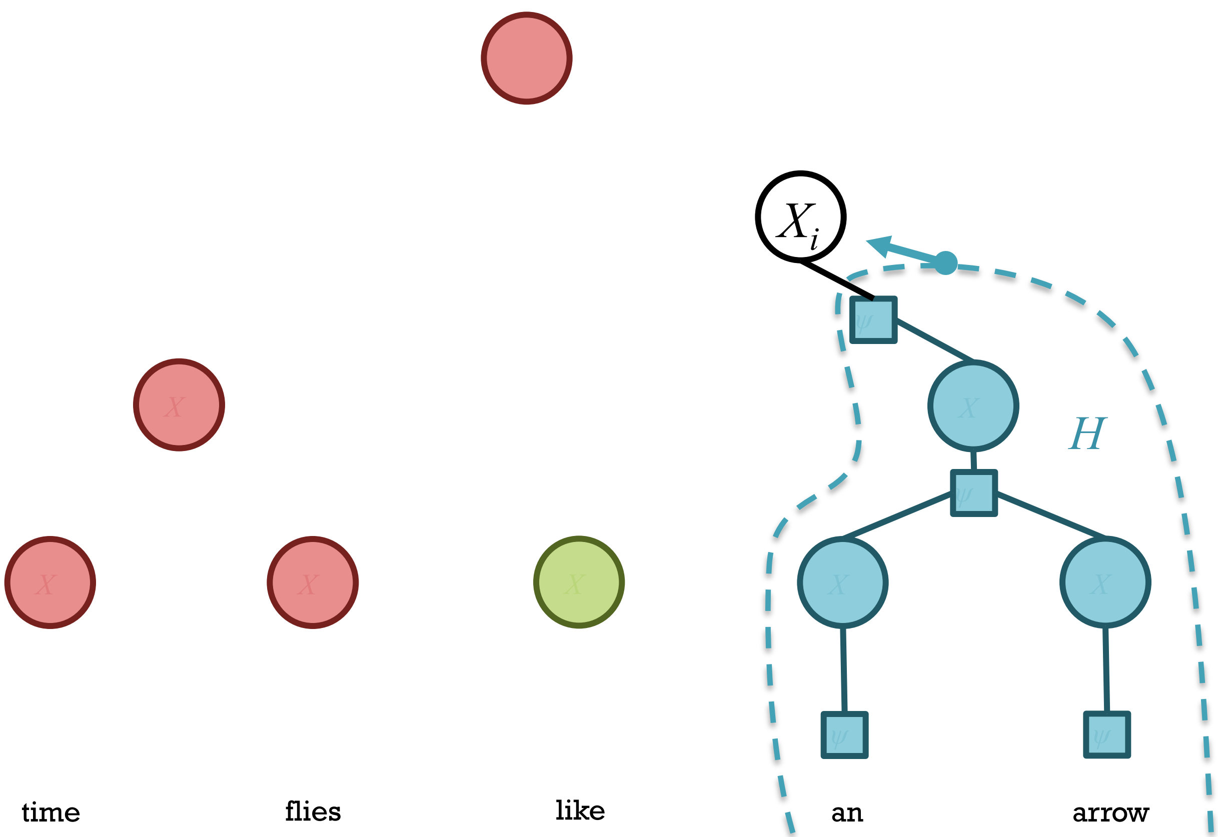
Subproblem:
Inference using just the factors in subgraph H

Figure adapted from Burkett & Klein (2012)

Acyclic BP as Dynamic Programming

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

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



Subproblem:
Inference using just the factors in subgraph H

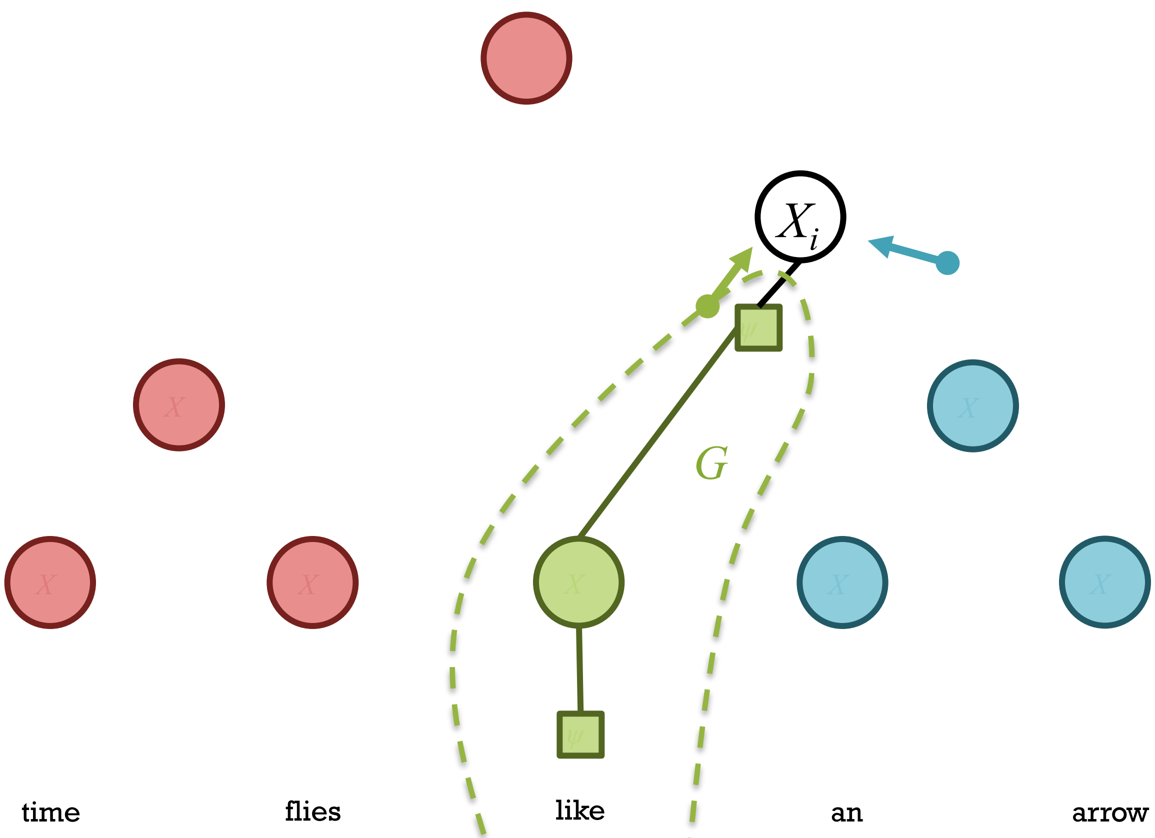
The marginal of X_i in that smaller model is the message sent to X_i from subgraph H

*Message **to** a variable*

Acyclic BP as Dynamic Programming

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

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



Subproblem:
Inference using just the factors in subgraph H

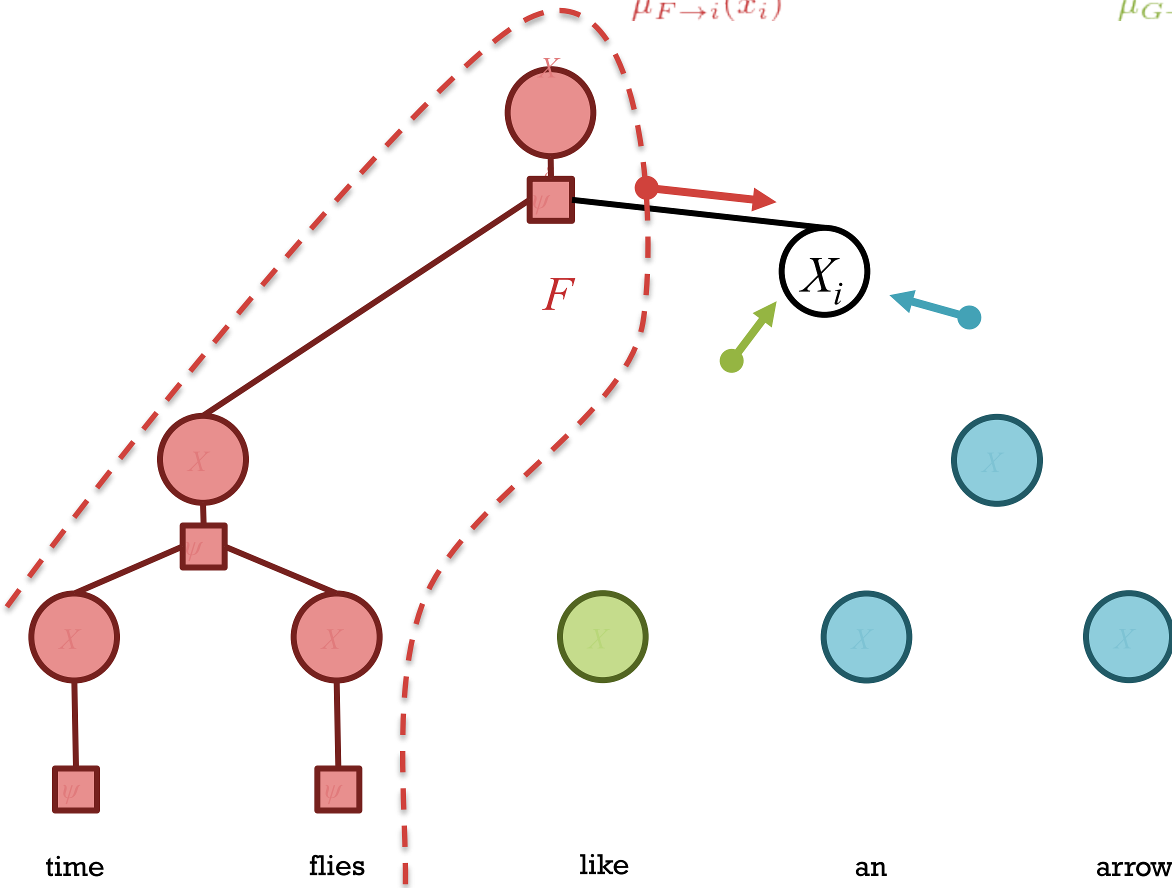
The marginal of X_i in that smaller model is the message sent to X_i from subgraph H

*Message **to** a variable*

Acyclic BP as Dynamic Programming

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

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



Subproblem:
Inference using just the factors in subgraph H

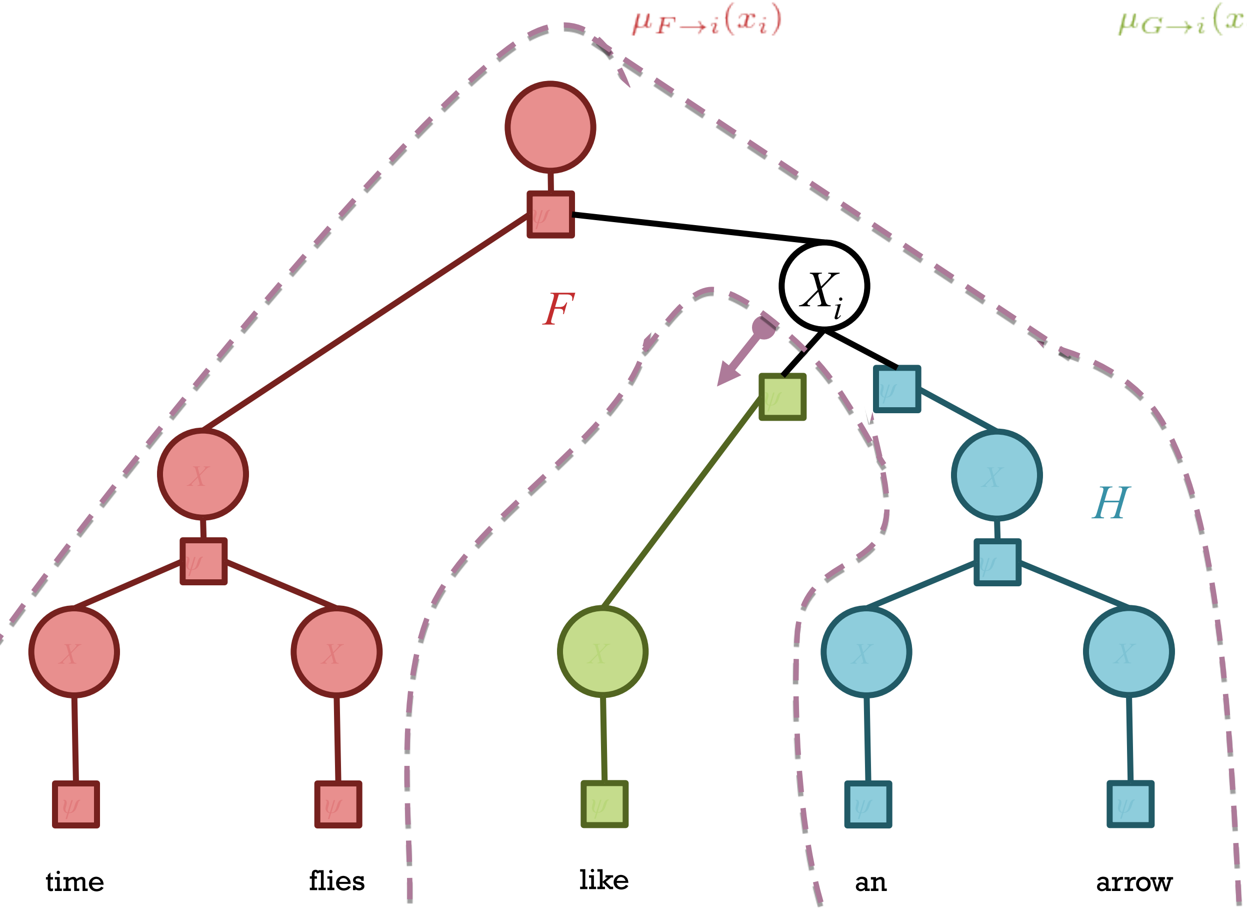
The marginal of X_i in that smaller model is the message sent to X_i from subgraph H

*Message **to** a variable*

Acyclic BP as Dynamic Programming

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

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



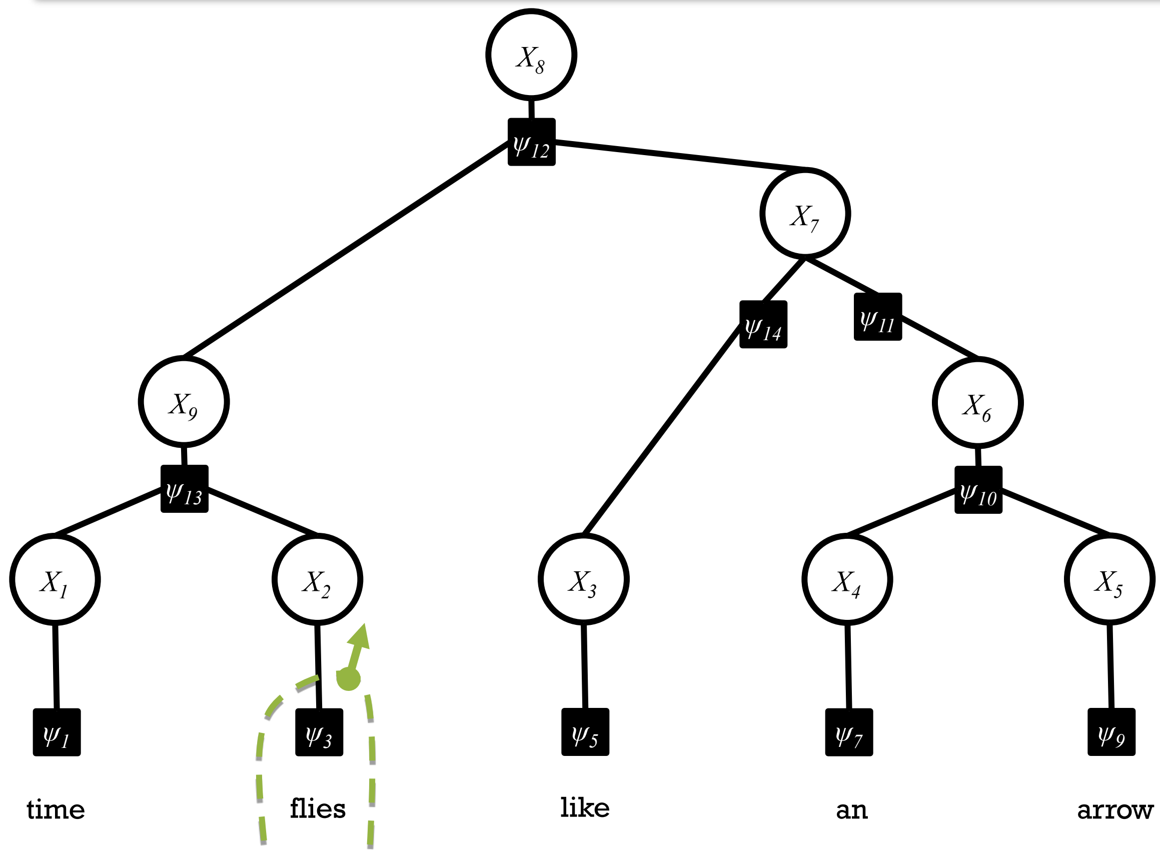
Subproblem:
Inference using just the factors in subgraph $F \cup H$

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

***Message from** a variable*

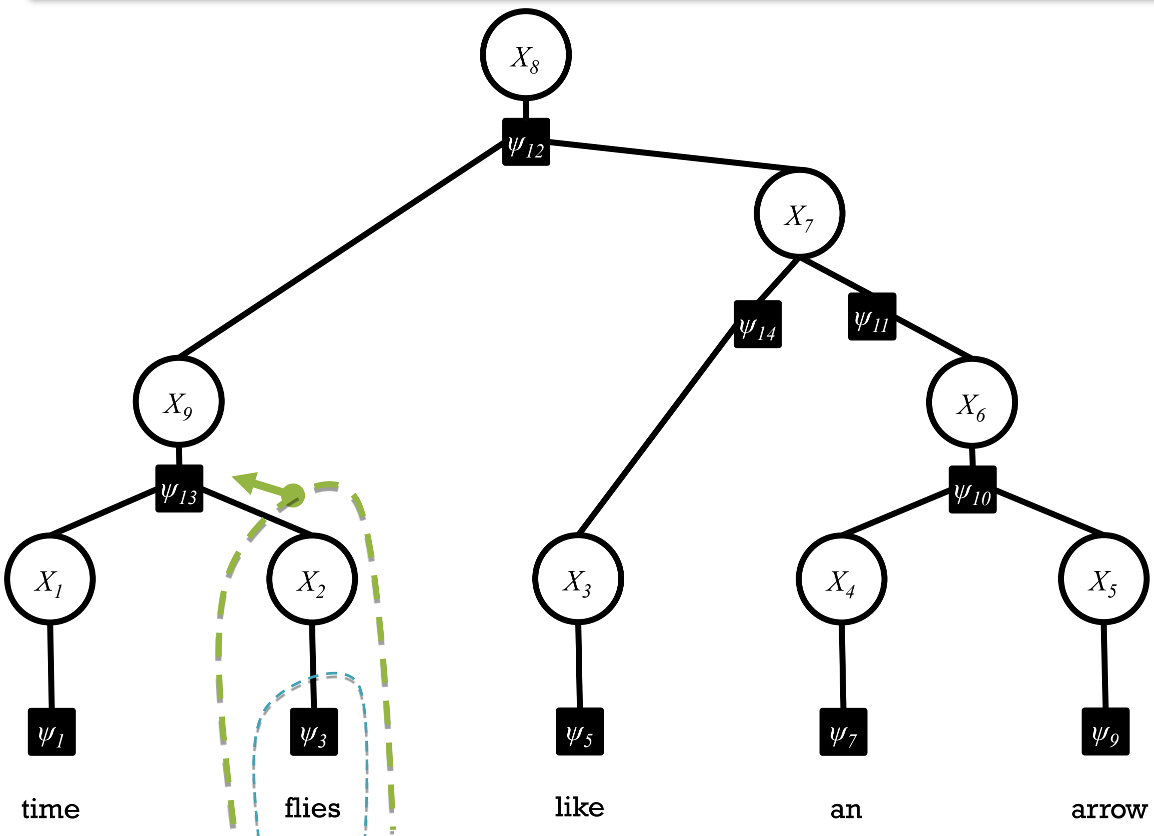
Acyclic BP as Dynamic Programming

- 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.



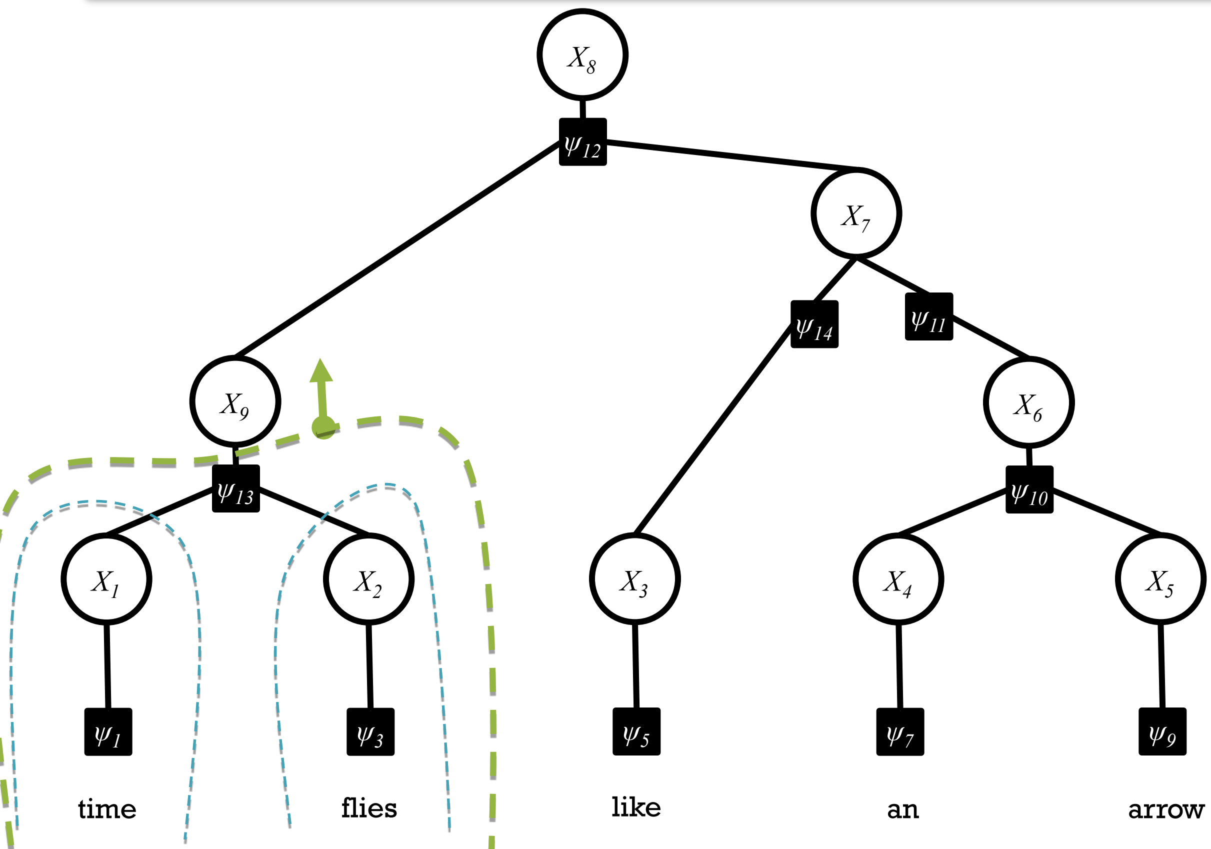
Acyclic BP as Dynamic Programming

- 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.



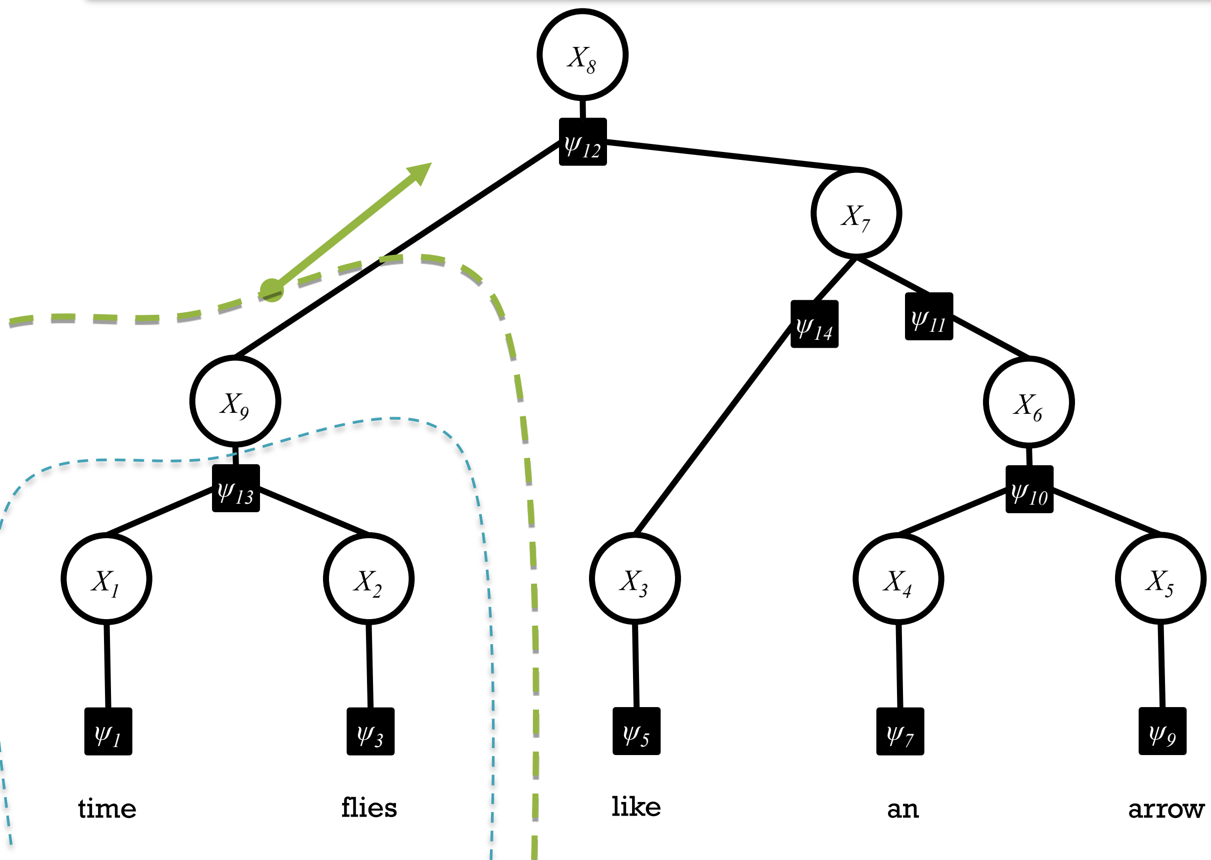
Acyclic BP as Dynamic Programming

- 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.



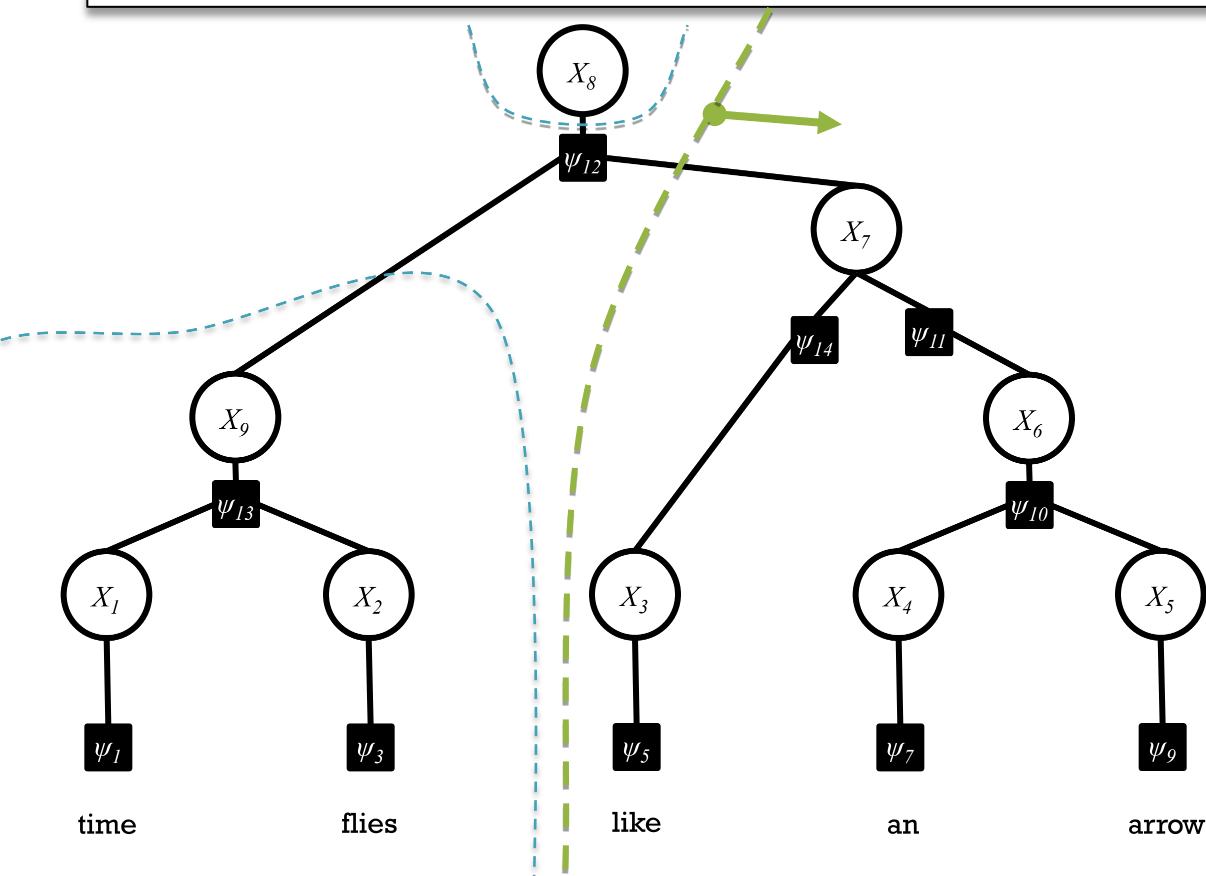
Acyclic BP as Dynamic Programming

- 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.



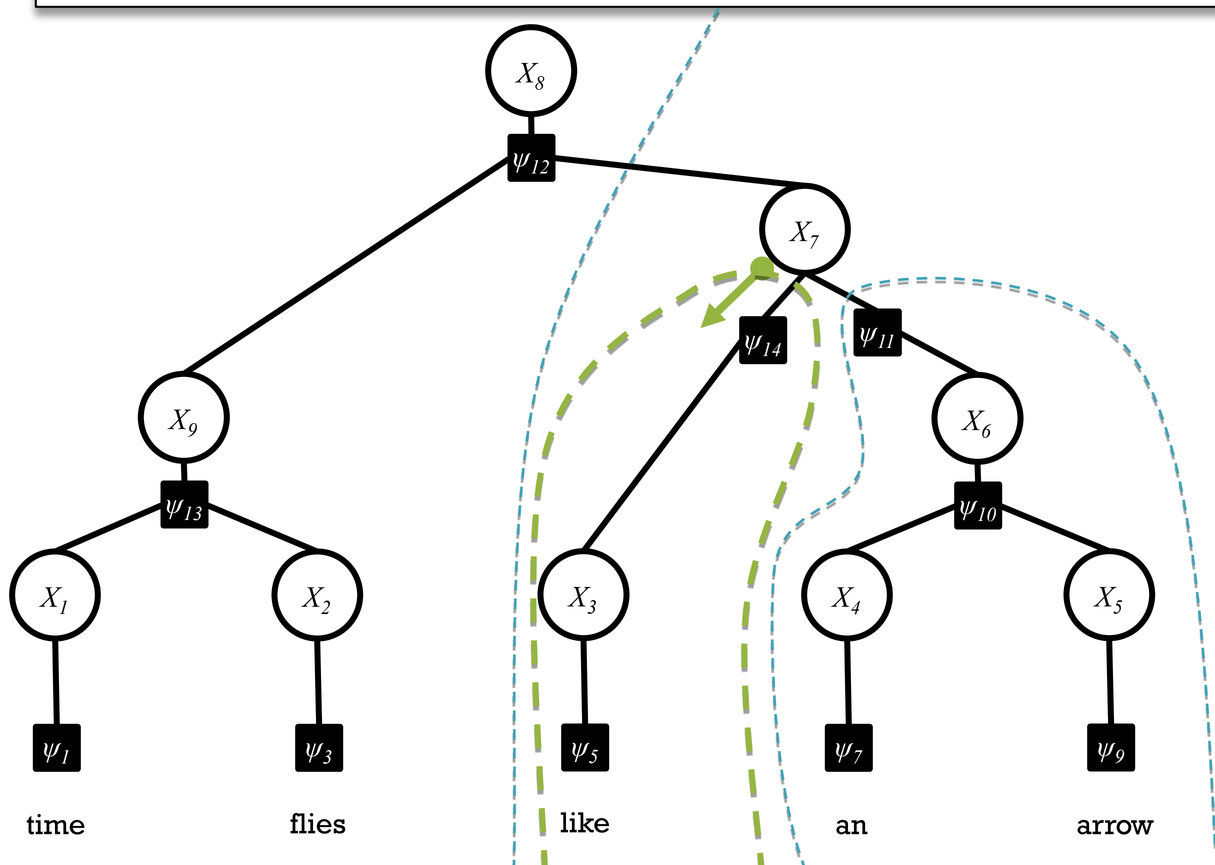
Acyclic BP as Dynamic Programming

- 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.



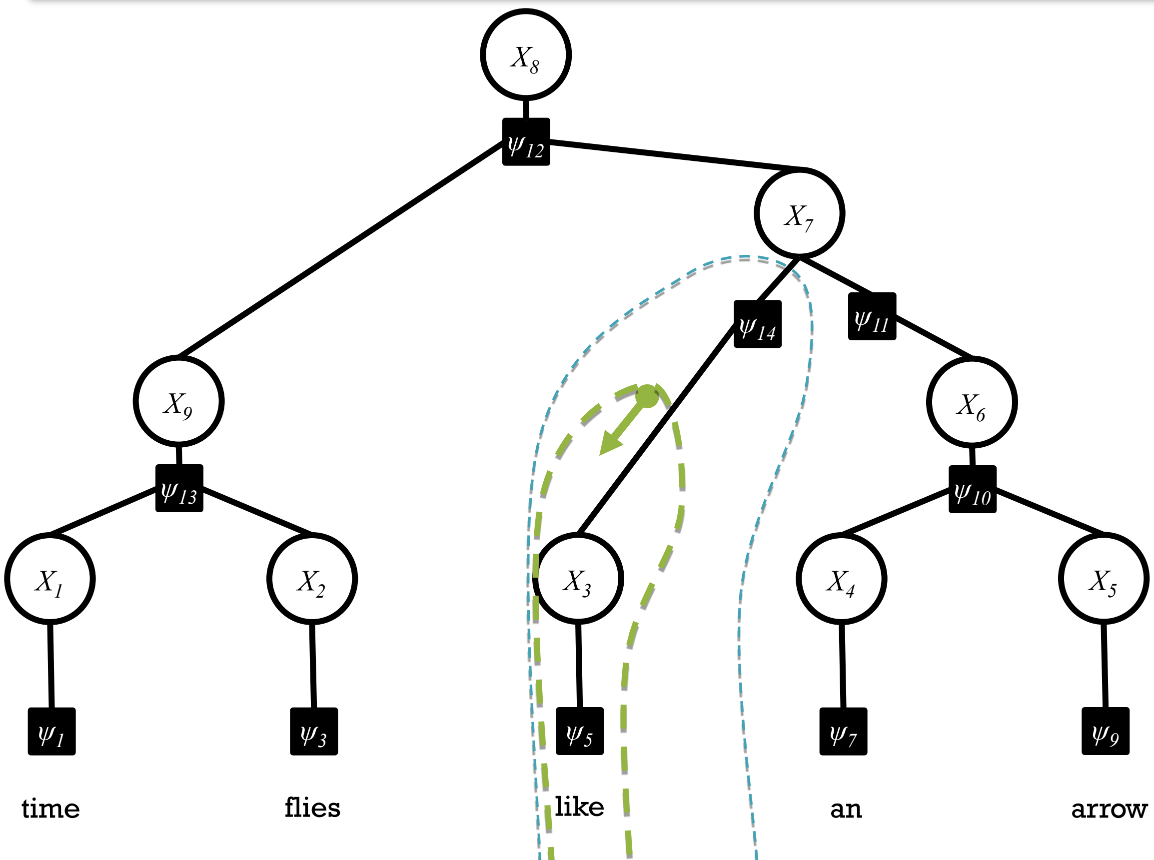
Acyclic BP as Dynamic Programming

- 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, $\boldsymbol{X}^* = \mathrm{argmax}_{\boldsymbol{X}}\, p(\boldsymbol{X})$

Max-product Belief Propagation

- Change the sum to a max:

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

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

- **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:

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

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

- **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:

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

Deterministic Annealing

Motivation: Smoothly transition from sum-product to max-product

1. Incorporate inverse temperature parameter into each factor:

Annealed Joint Distribution

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

2. Send messages as usual for sum-product BP
3. Anneal T from 1 to 0:

$T = 1$	Sum-product
$T \rightarrow 0$	Max-product

4. Take resulting beliefs to power T

Semirings

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

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

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

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

Exact inference for linear chain models

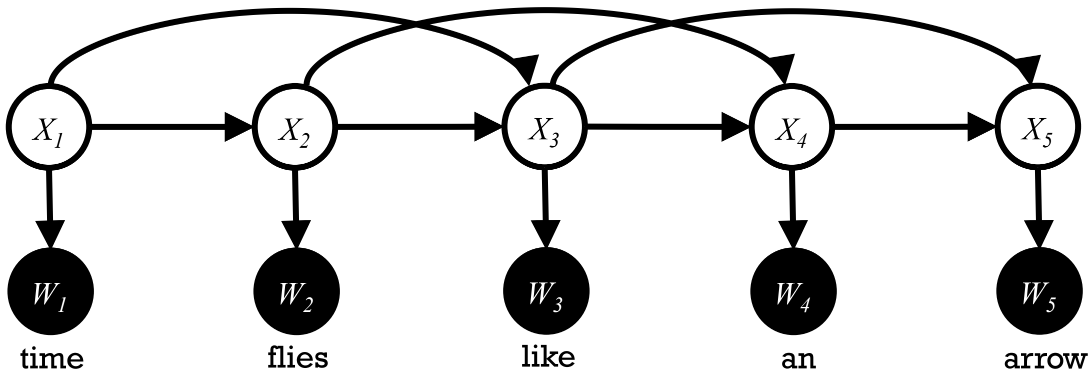
FORWARD-BACKWARD AND VITERBI ALGORITHMS

Forward-Backward Algorithm

- Sum-product BP on an HMM is called the **forward-backward algorithm**
- Max-product BP on an HMM is called the **Viterbi algorithm**

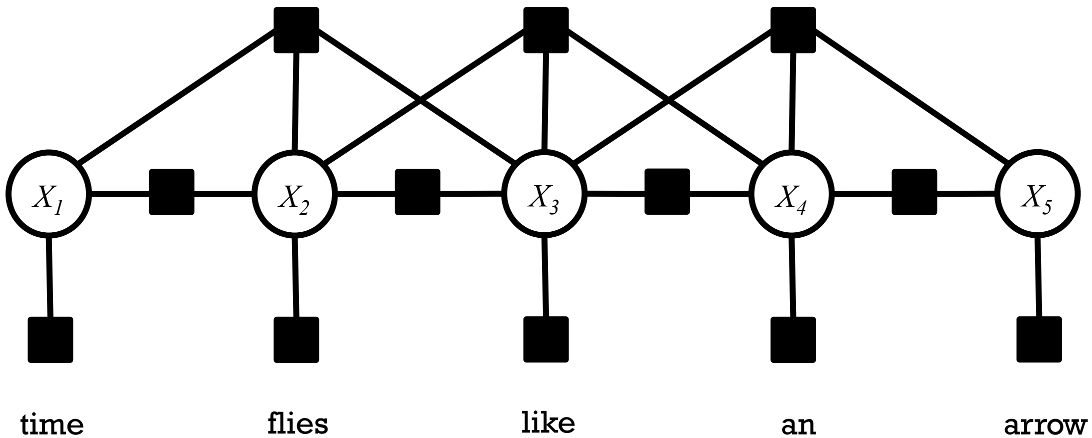
Forward-Backward Algorithm

Trigram HMM is not a tree, even when converted to a factor graph



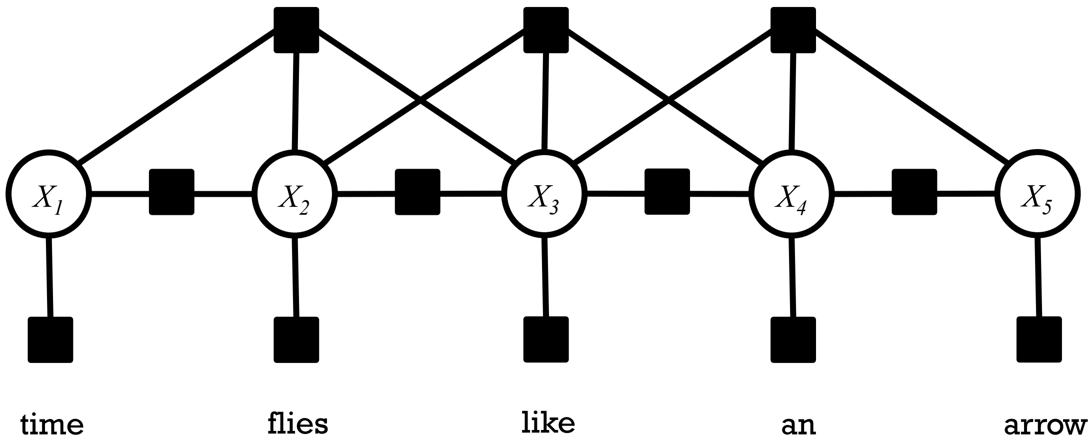
Forward-Backward Algorithm

Trigram HMM is not a tree, even when converted to a factor graph



Forward-Backward Algorithm

Trigram HMM is not a tree, even when converted to a factor graph



Trick: (See also Sha & Pereira (2003))

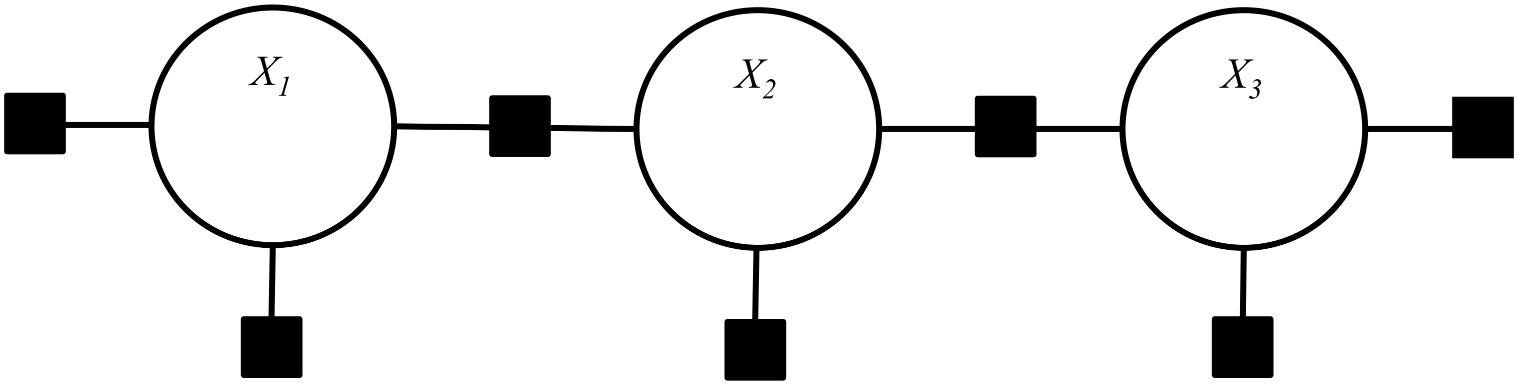
- Replace each variable domain with its cross product
  e.g. {B,I,O} → {BB, BI, BO, IB, II, IO, OB, OI, OO}
- Replace each pair of variables with a single one. For all i, $y_{i,i+1} = (x_i, x_{i+1})$
- Add features with weight -∞ that disallow illegal configurations between pairs of the new variables
  e.g. **legal** = BI and IO **illegal** = II and OO
- This is effectively a special case of the junction tree algorithm

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.

EXTRA SLIDES ON FORWARD BACKWARD AS SUM-PRODUCT BP

CRF Tagging Model



find

preferred

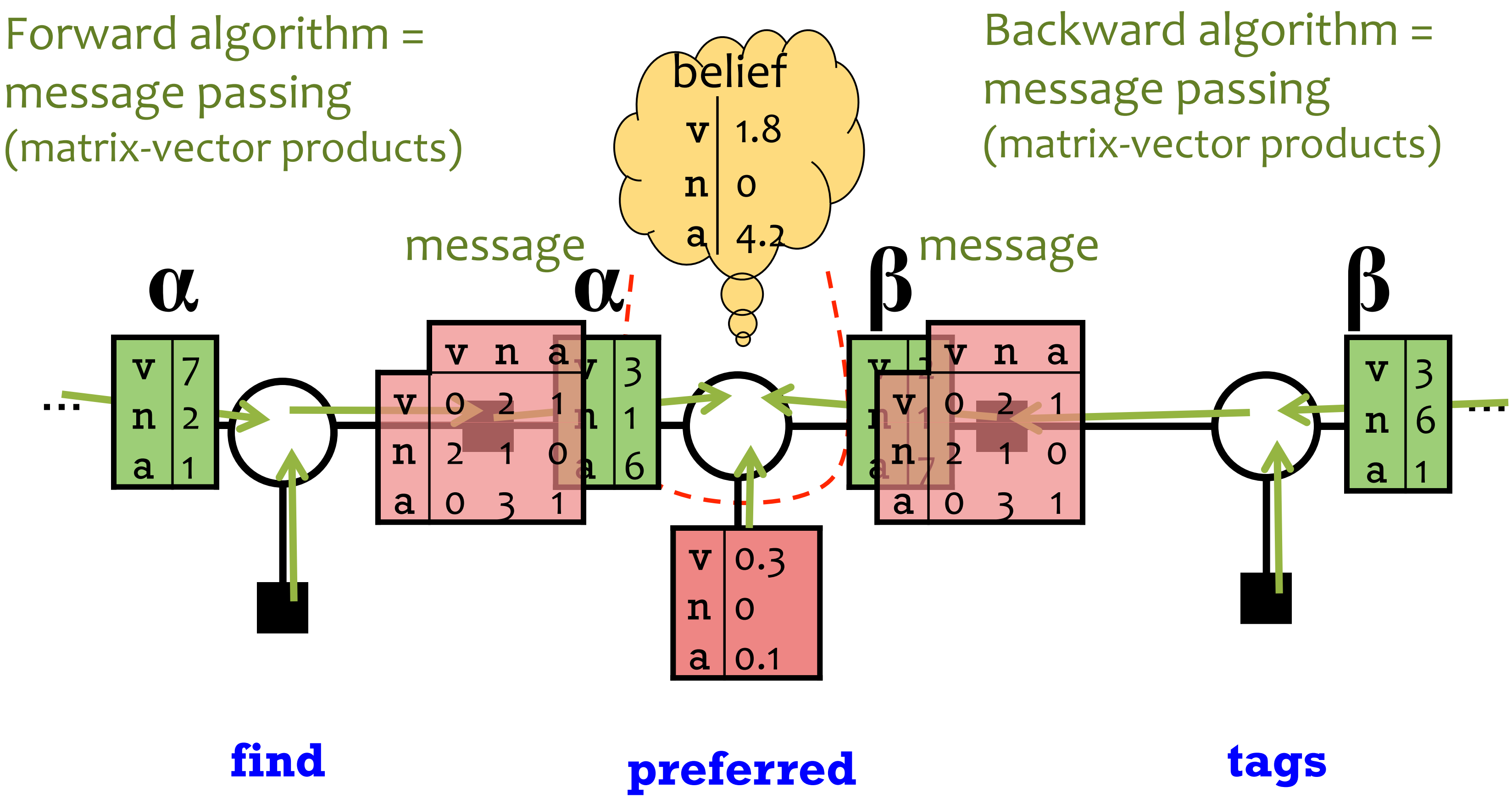
tags

Could be verb or noun

Could be adjective or verb

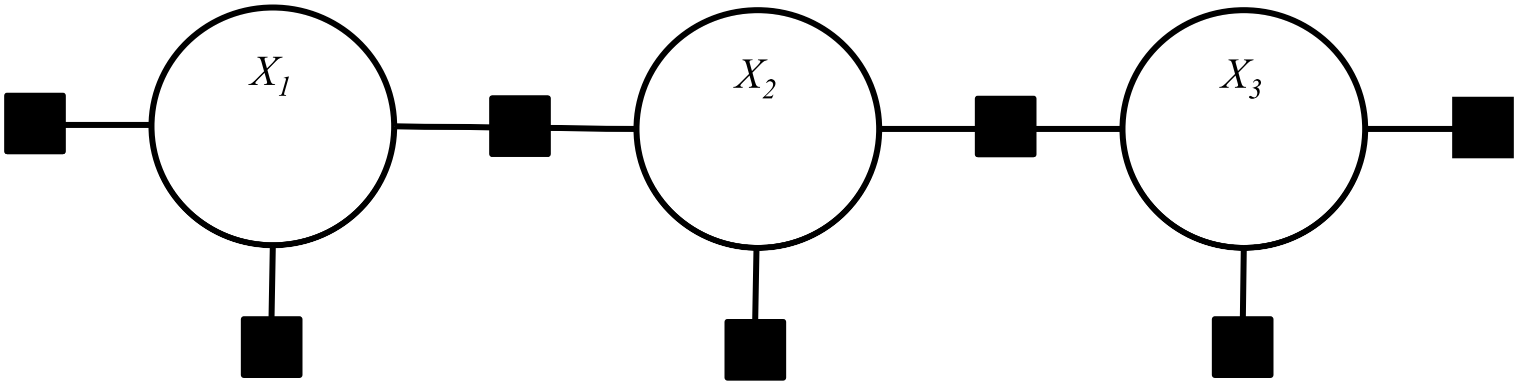
Could be noun or verb

CRF Tagging by Belief Propagation



- Forward-backward is a message passing algorithm.
- It’s the simplest case of belief propagation.

So Let's Review Forward-Backward …



find

preferred

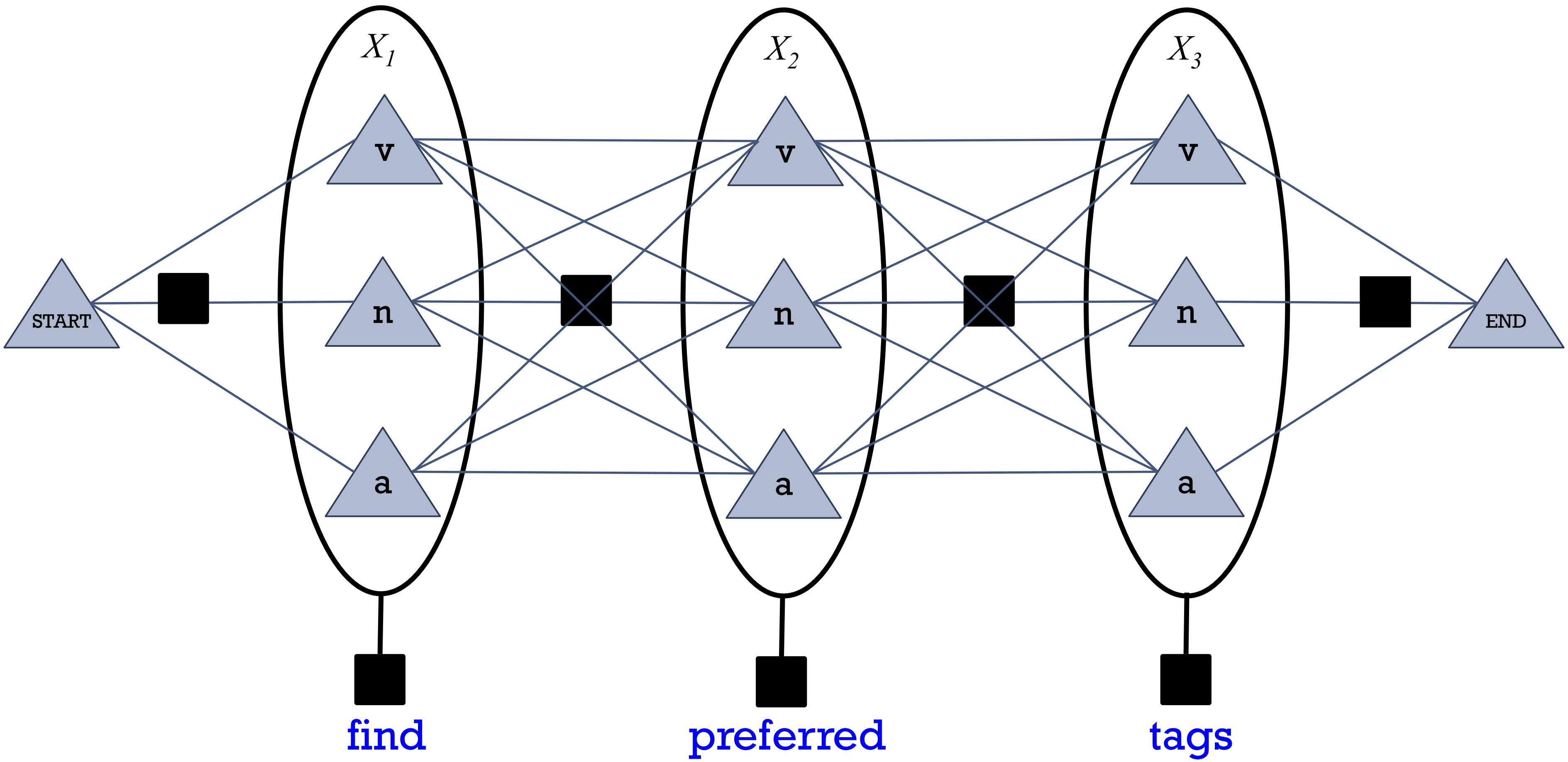
tags

Could be verb or noun

Could be adjective or verb

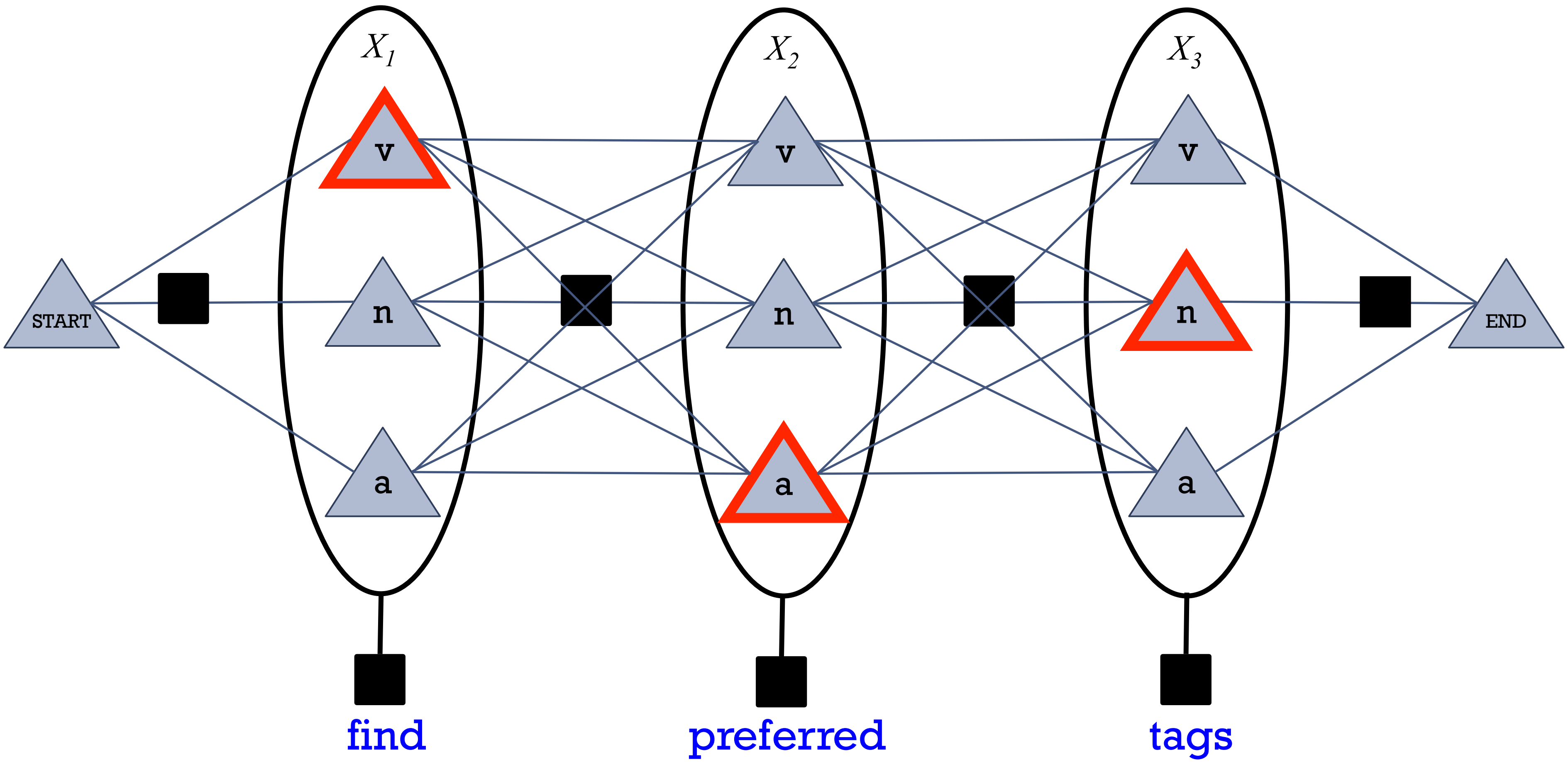
Could be noun or verb

So Let's Review Forward-Backward …



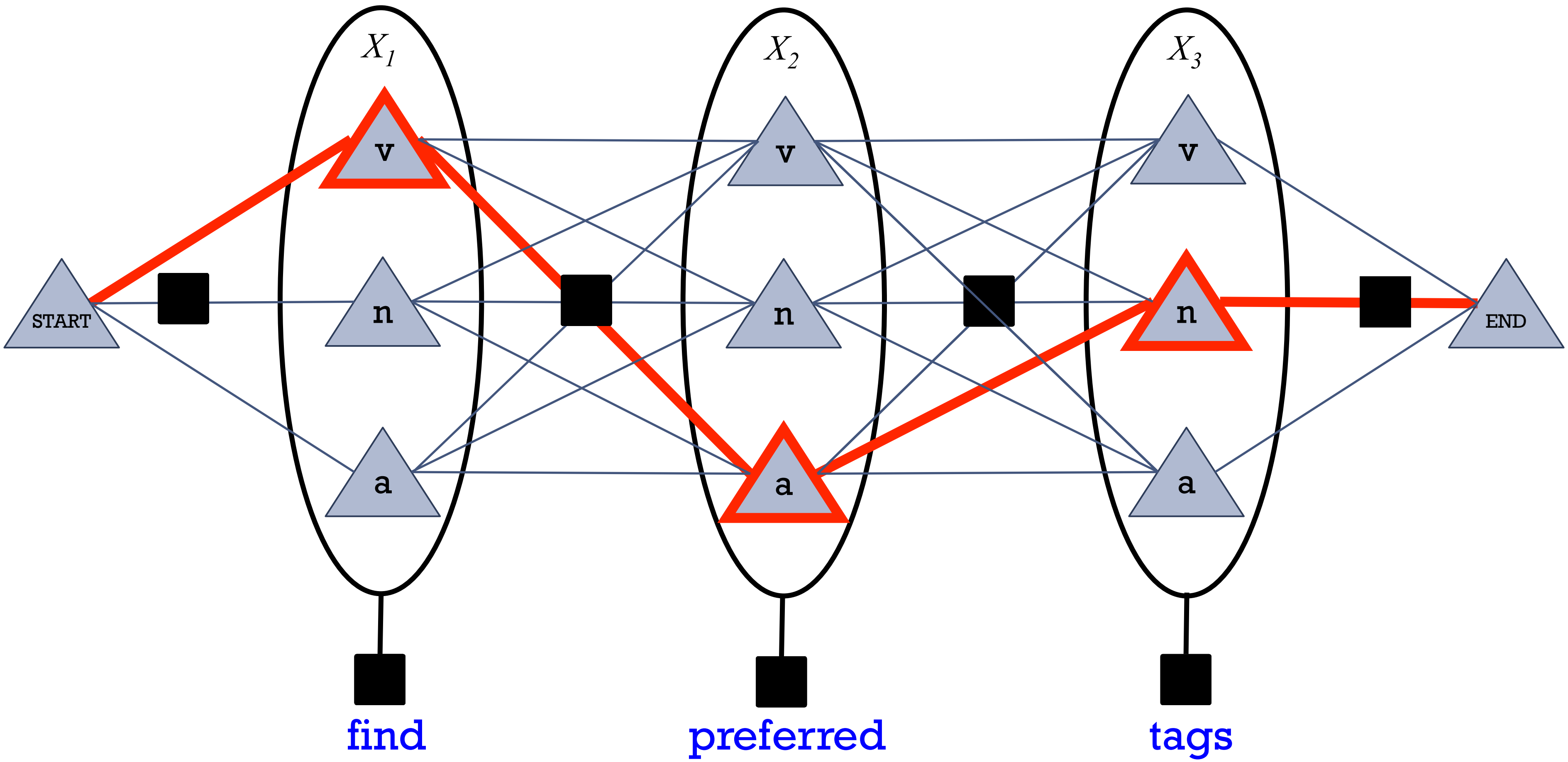
- Show the possible *values* for each variable

So Let's Review Forward-Backward …



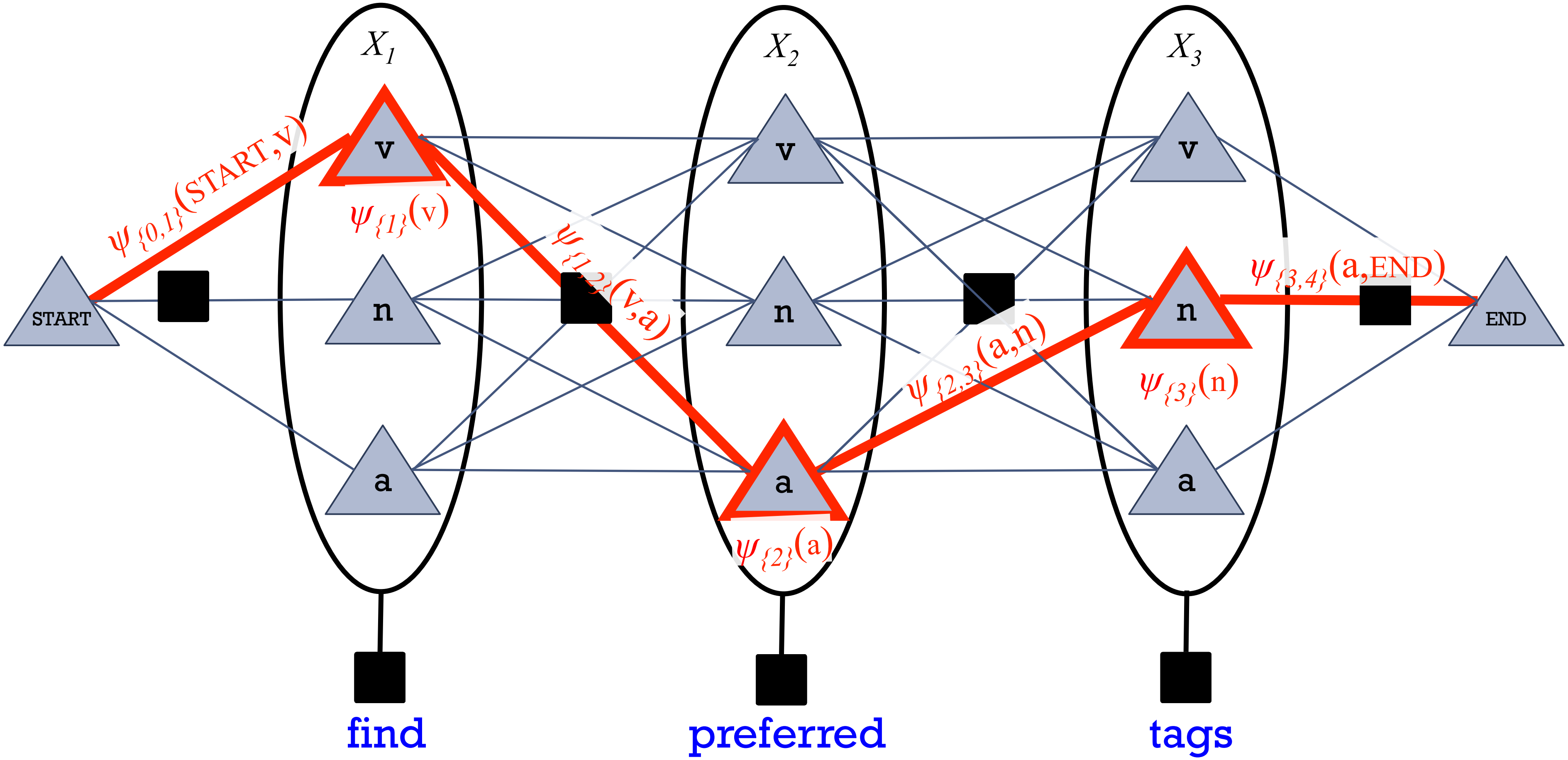
- Let's show the possible *values* for each variable
- One possible assignment

So Let's Review Forward-Backward …



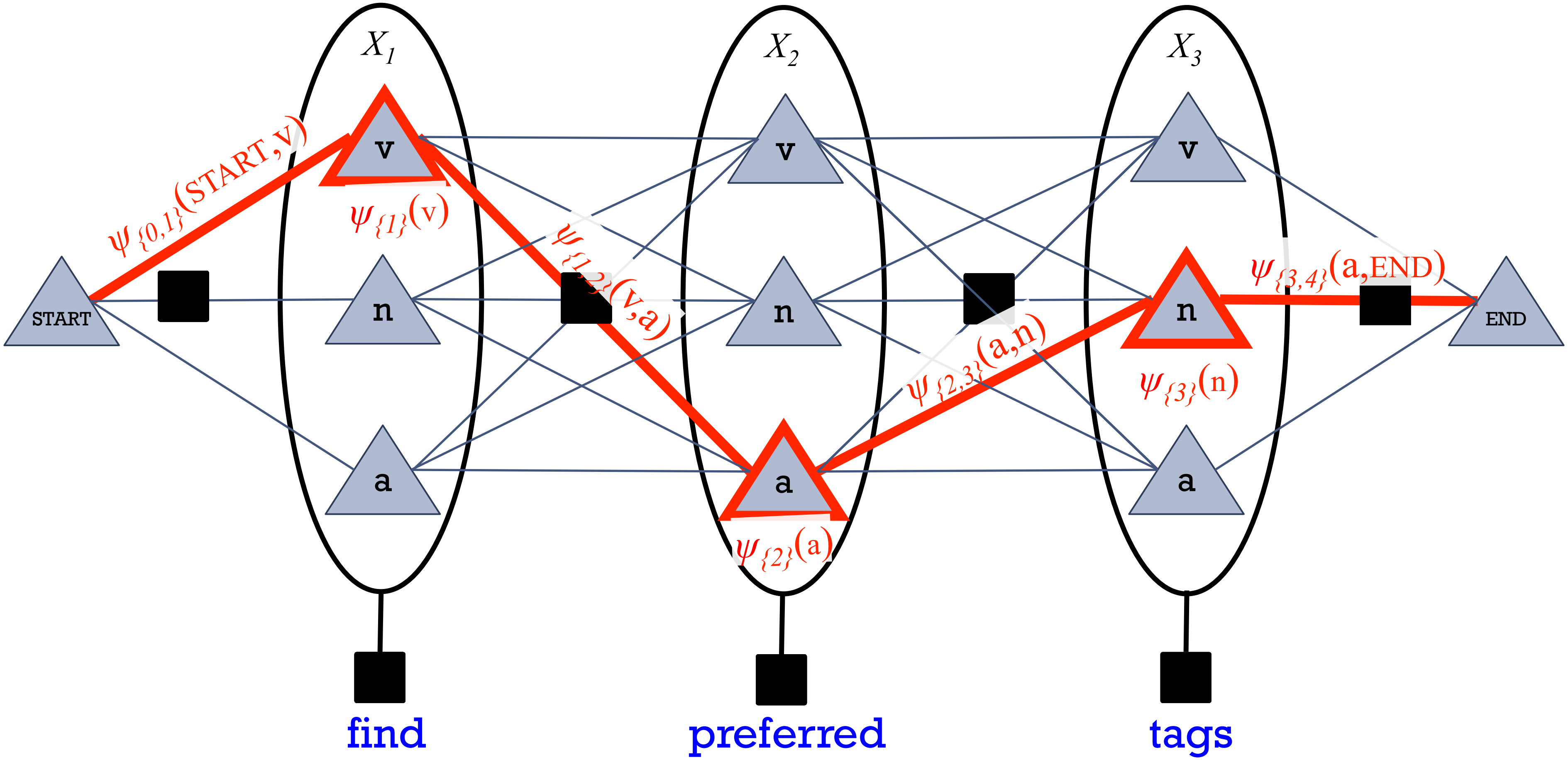
- Let's show the possible *values* for each variable
- One possible assignment
- And what the 7 factors think of it …

Viterbi Algorithm: Most Probable Assignment



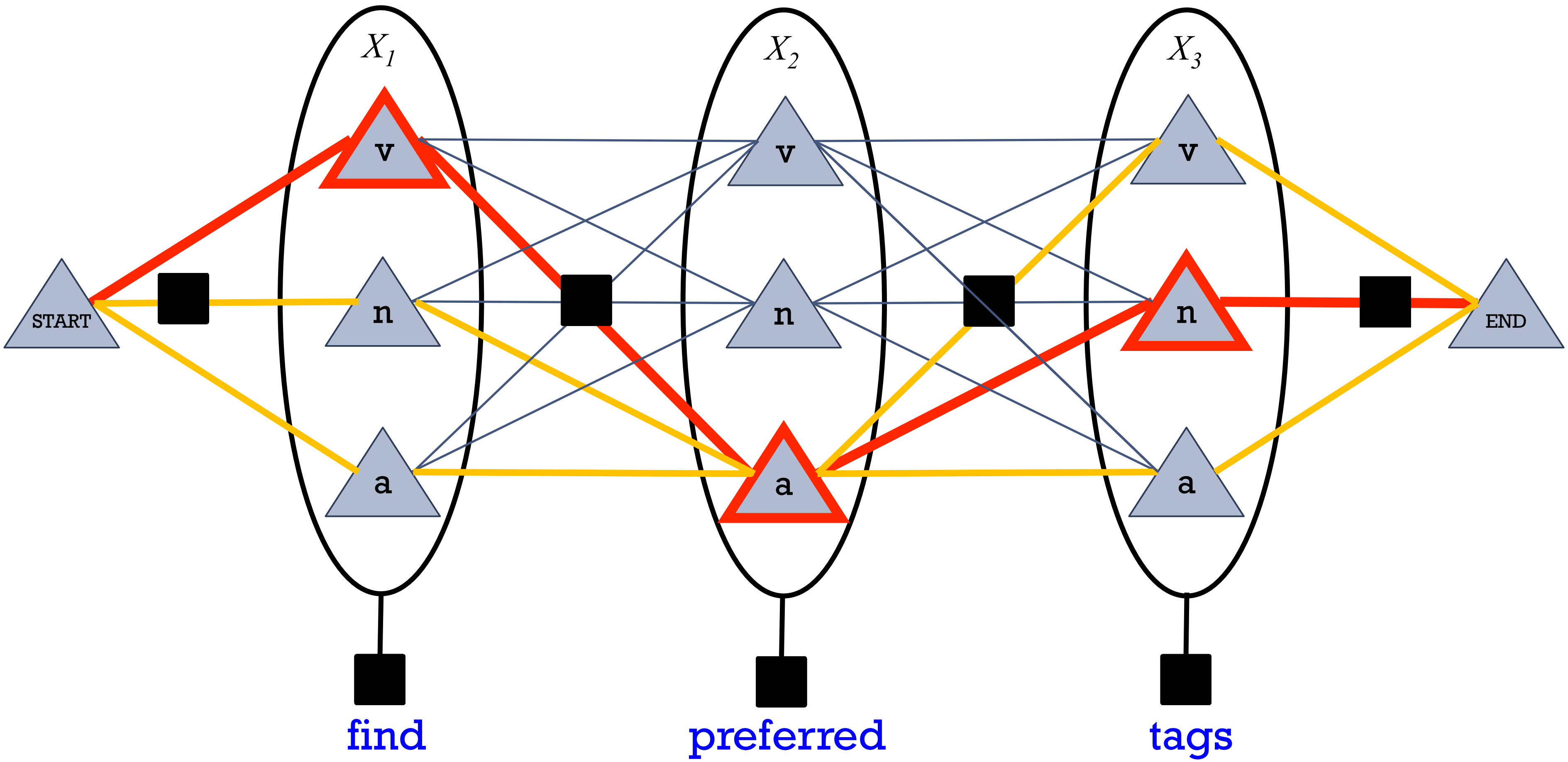
- So p(**v a n**) = (1/Z) * product of 7 numbers
- Numbers associated with edges and nodes of path
- Most probable assignment = **path with highest product**

Viterbi Algorithm: Most Probable Assignment



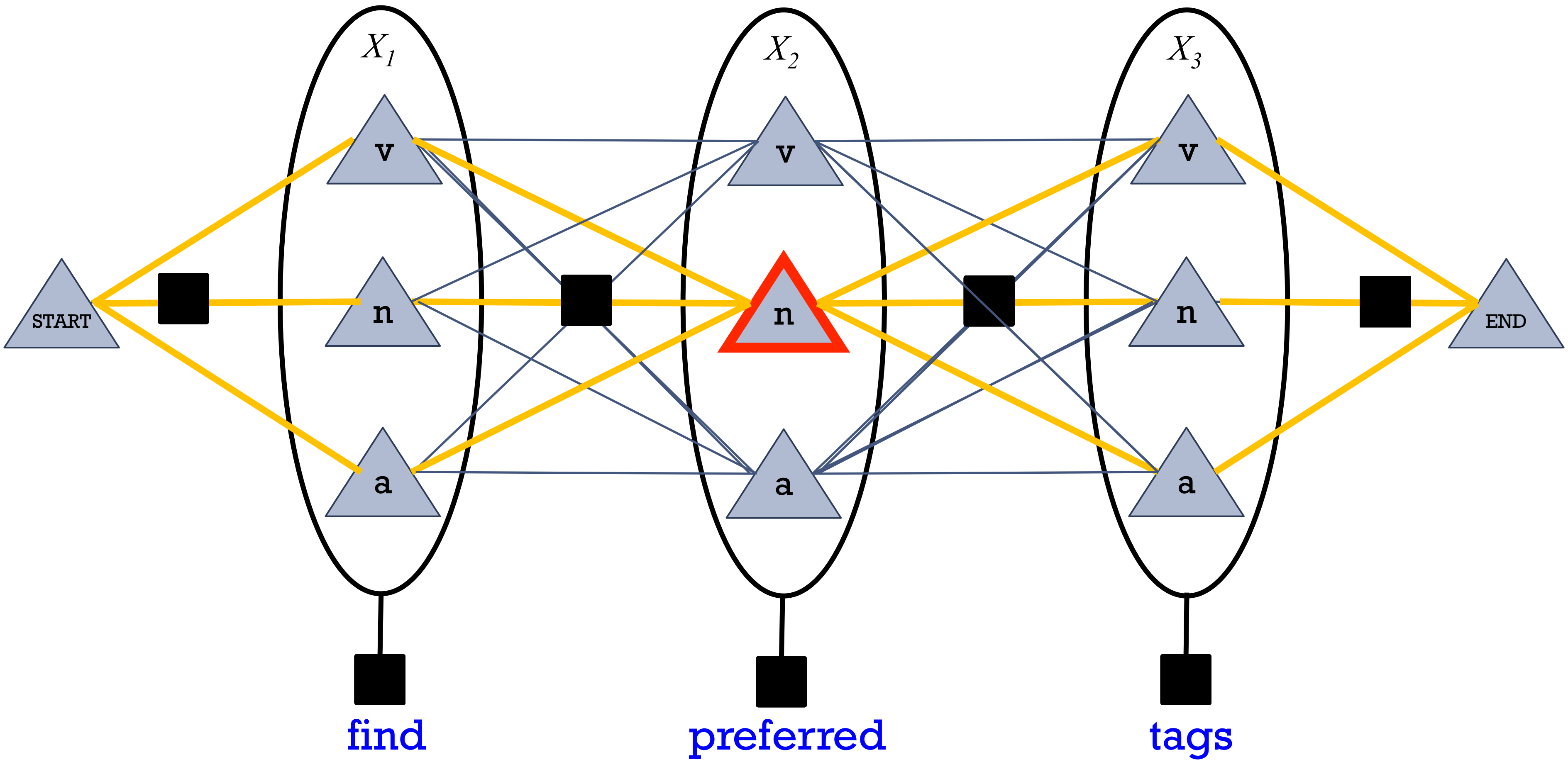
- So p(**v a n**) = (1/Z) * product weight of one path

Forward-Backward Algorithm: Finds Marginals



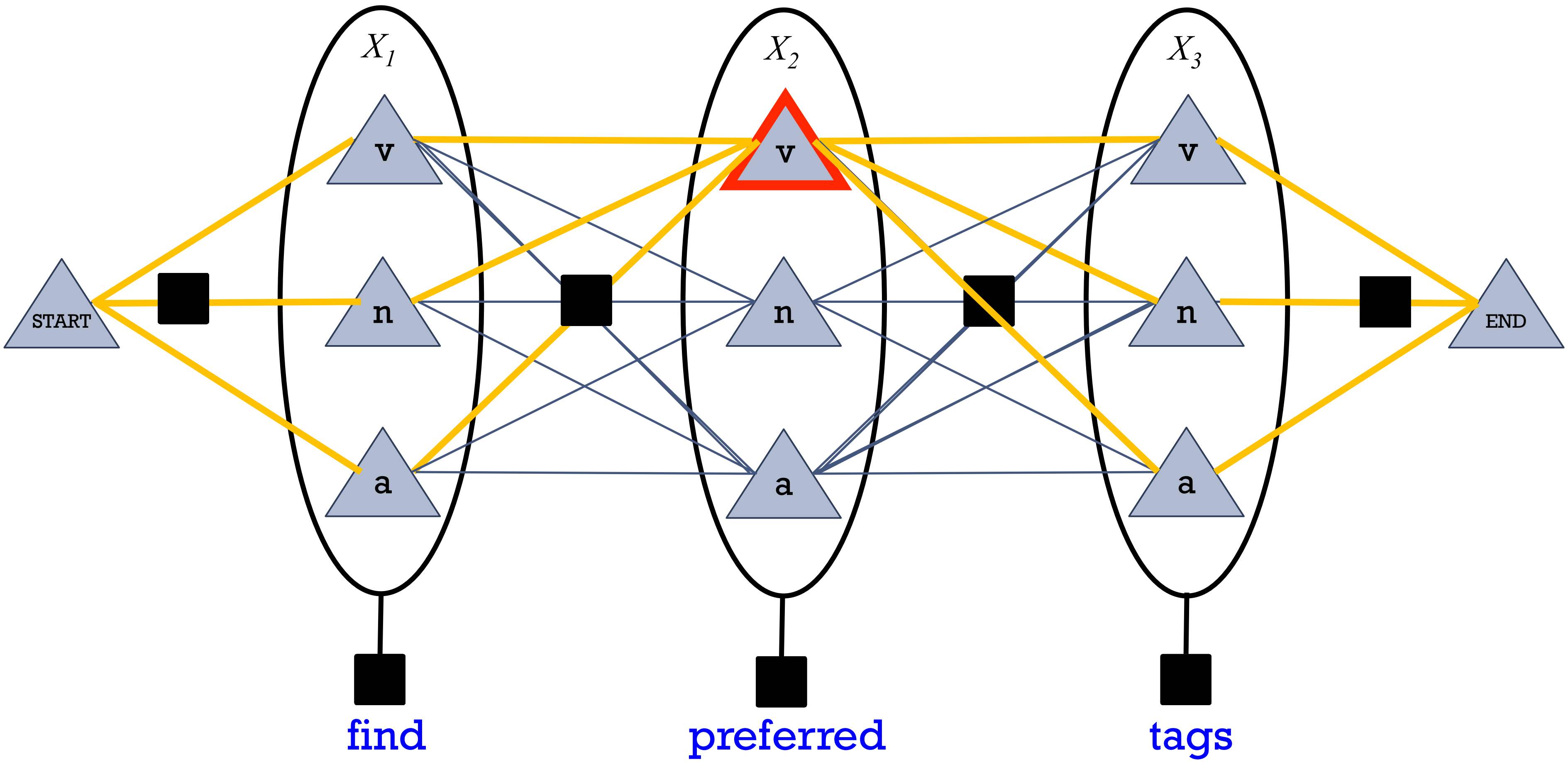
- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability $p(X_2 = a)$
  = (1/Z) * total weight of *all* paths through a

Forward-Backward Algorithm: Finds Marginals



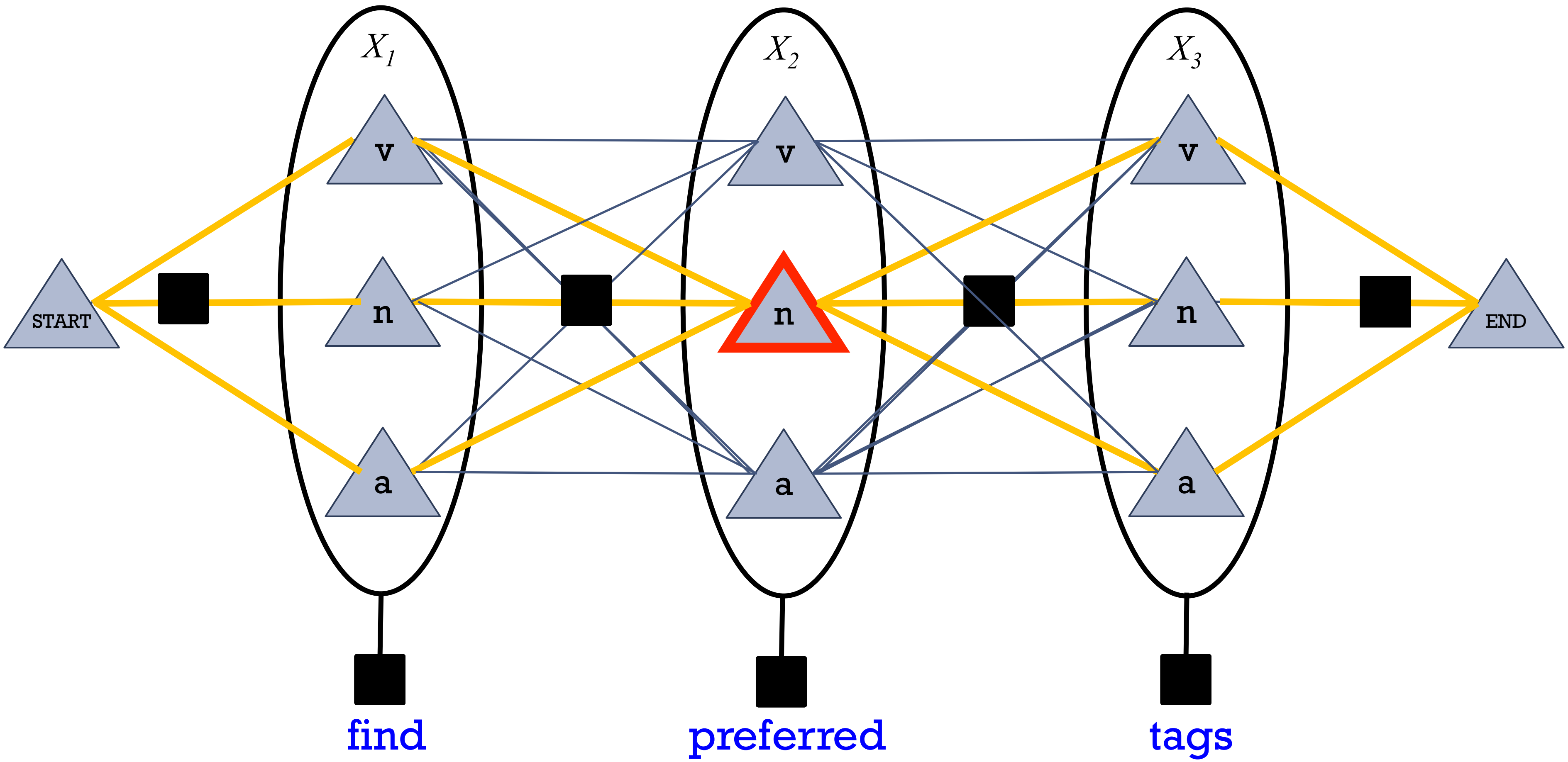
- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability $p(X_2 = a)$
  = (1/Z) * total weight of *all* paths through **n**

Forward-Backward Algorithm: Finds Marginals



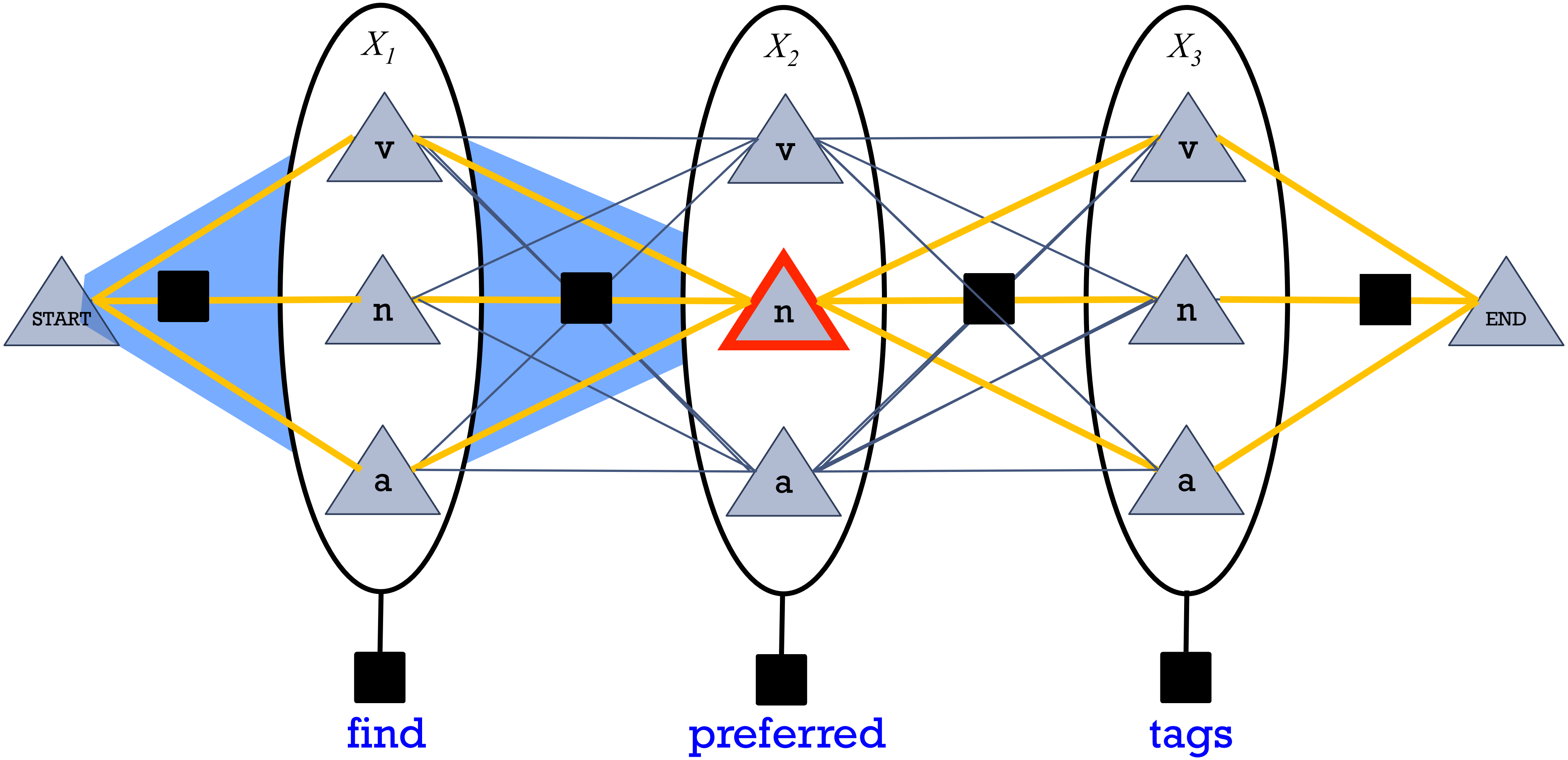
- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability $p(X_2 = a)$
  = (1/Z) * total weight of *all* paths through v

Forward-Backward Algorithm: Finds Marginals



- So p(**v a n**) = (1/Z) * product weight of one path
- Marginal probability p(X_2 = a)
  = (1/Z) * total weight of *all* paths through **n**

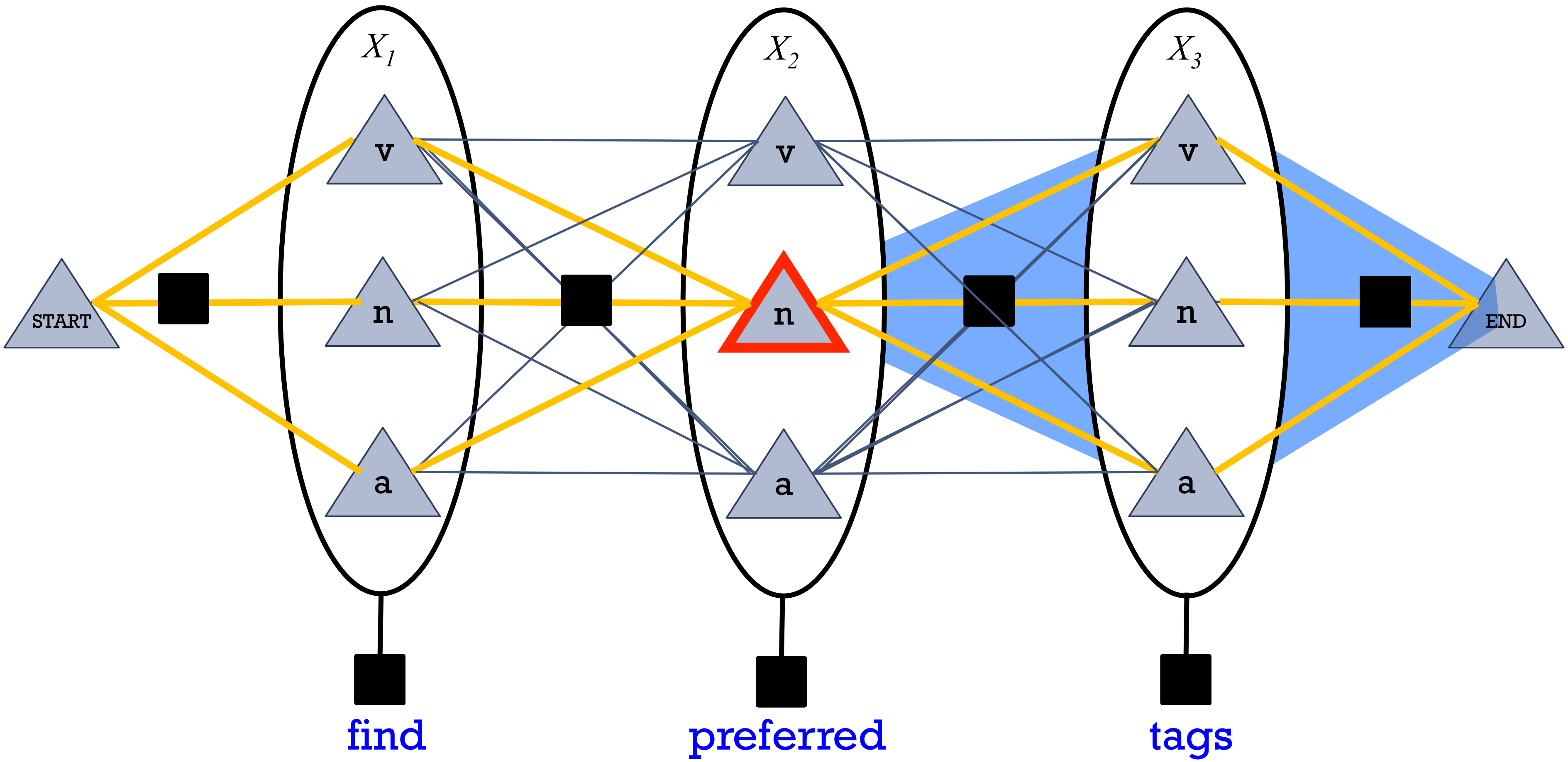
Forward-Backward Algorithm: Finds Marginals



$\alpha_2(\mathbf{n})$ = total weight of these path *prefixes*

(found by dynamic programming: matrix-vector products)

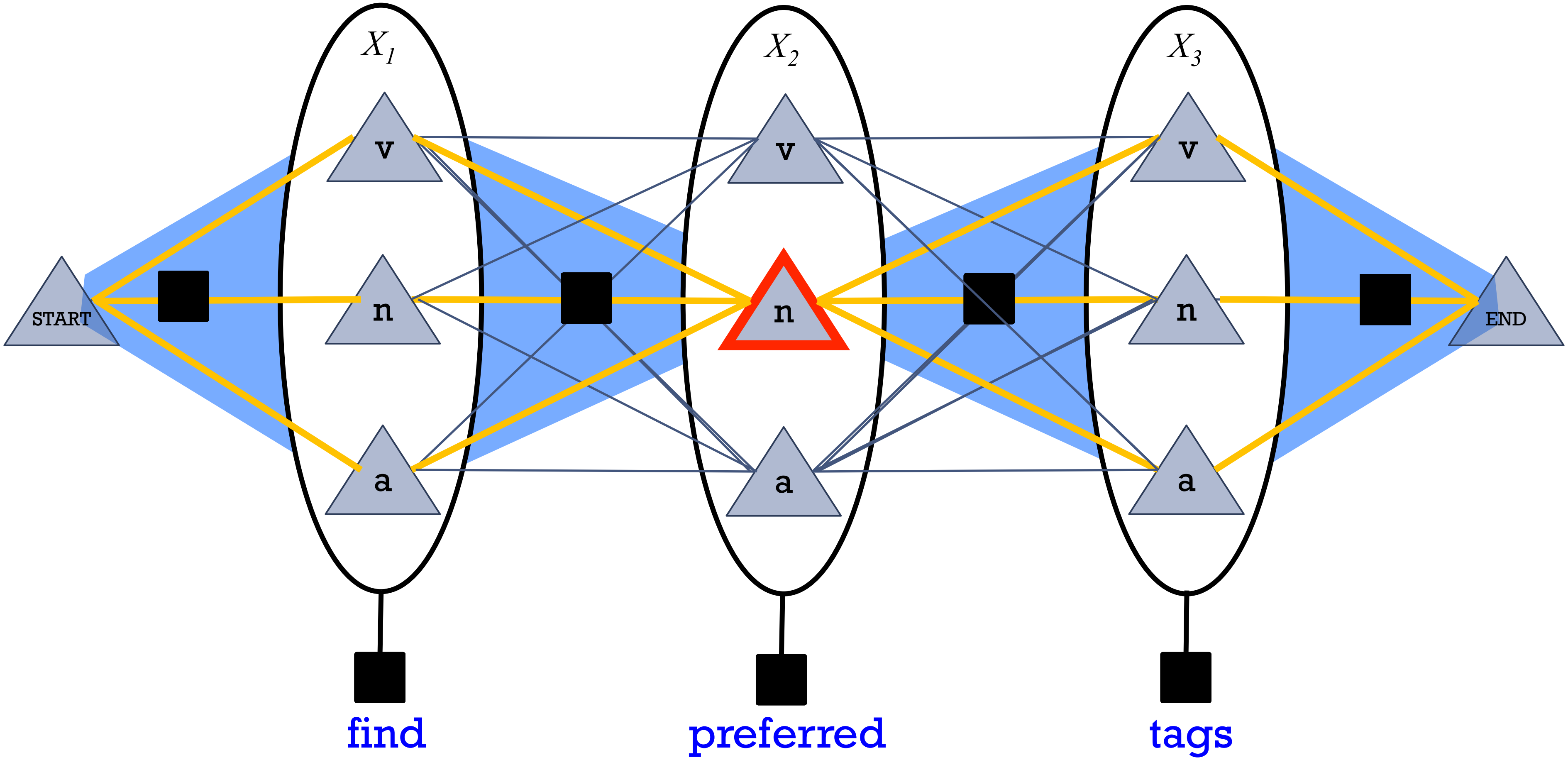
Forward-Backward Algorithm: Finds Marginals



$\beta_2(\mathbf{n})$ = total weight of these path *suffixes*

(found by dynamic programming: matrix-vector products)

Forward-Backward Algorithm: Finds Marginals



$\alpha_2(\mathbf{n})$ = total weight of these path *prefixes* (a + b + c)

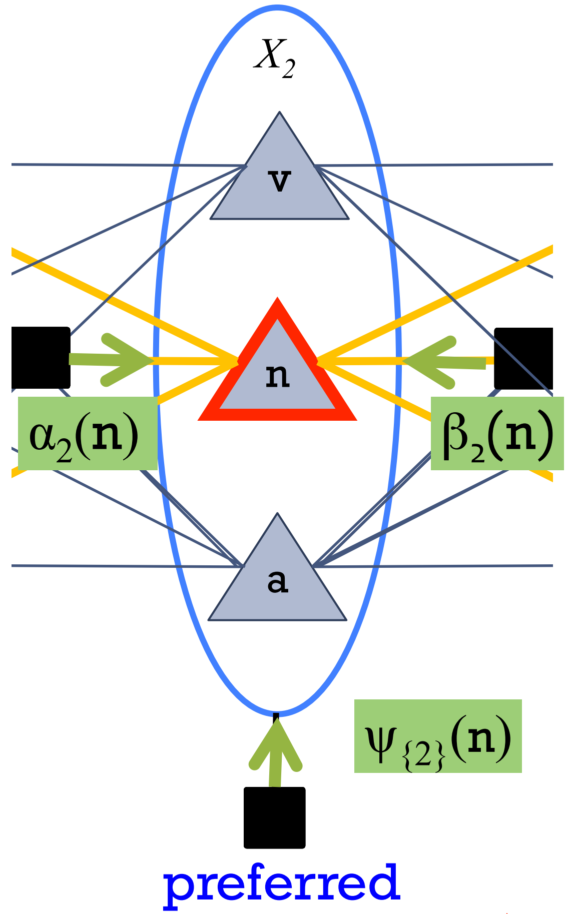
$\beta_2(\mathbf{n})$ = total weight of these path *suffixes* (x + y + z)

Product gives ax+ay+az+bx+by+bz+cx+cy+cz = total weight of paths

Forward-Backward Algorithm: Finds Marginals

Oops! The weight of a path through a state also includes a weight at that state.
So $\alpha(\mathbf{n})\cdot\beta(\mathbf{n})$ isn't enough.

The extra weight is the opinion of the unigram factor at this variable.

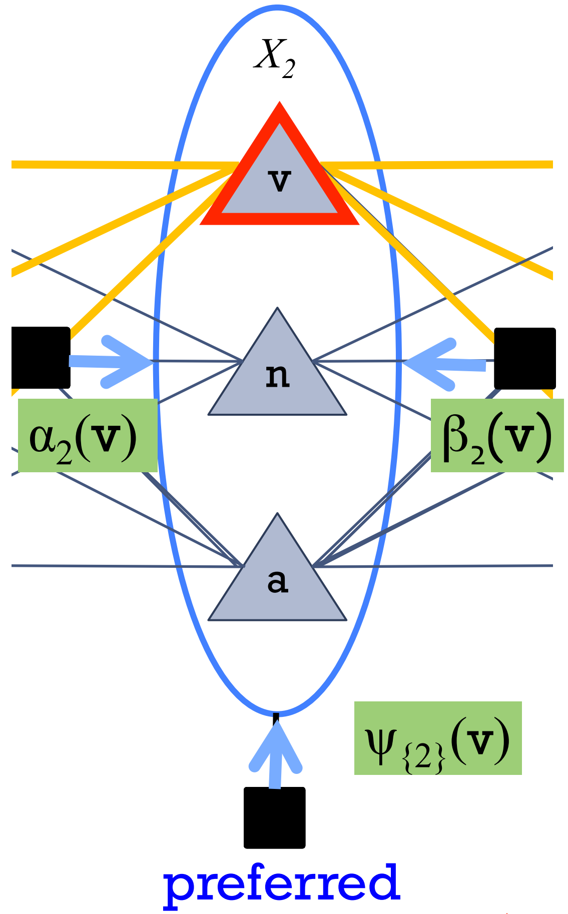


"belief that $X_2 = \mathbf{n}$"

total weight of *all* paths through n

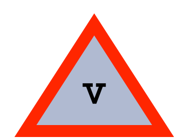
$= \alpha_2(\mathbf{n})\ \psi_{\{2\}}(\mathbf{n})\ \beta_2(\mathbf{n})$

Forward-Backward Algorithm: Finds Marginals



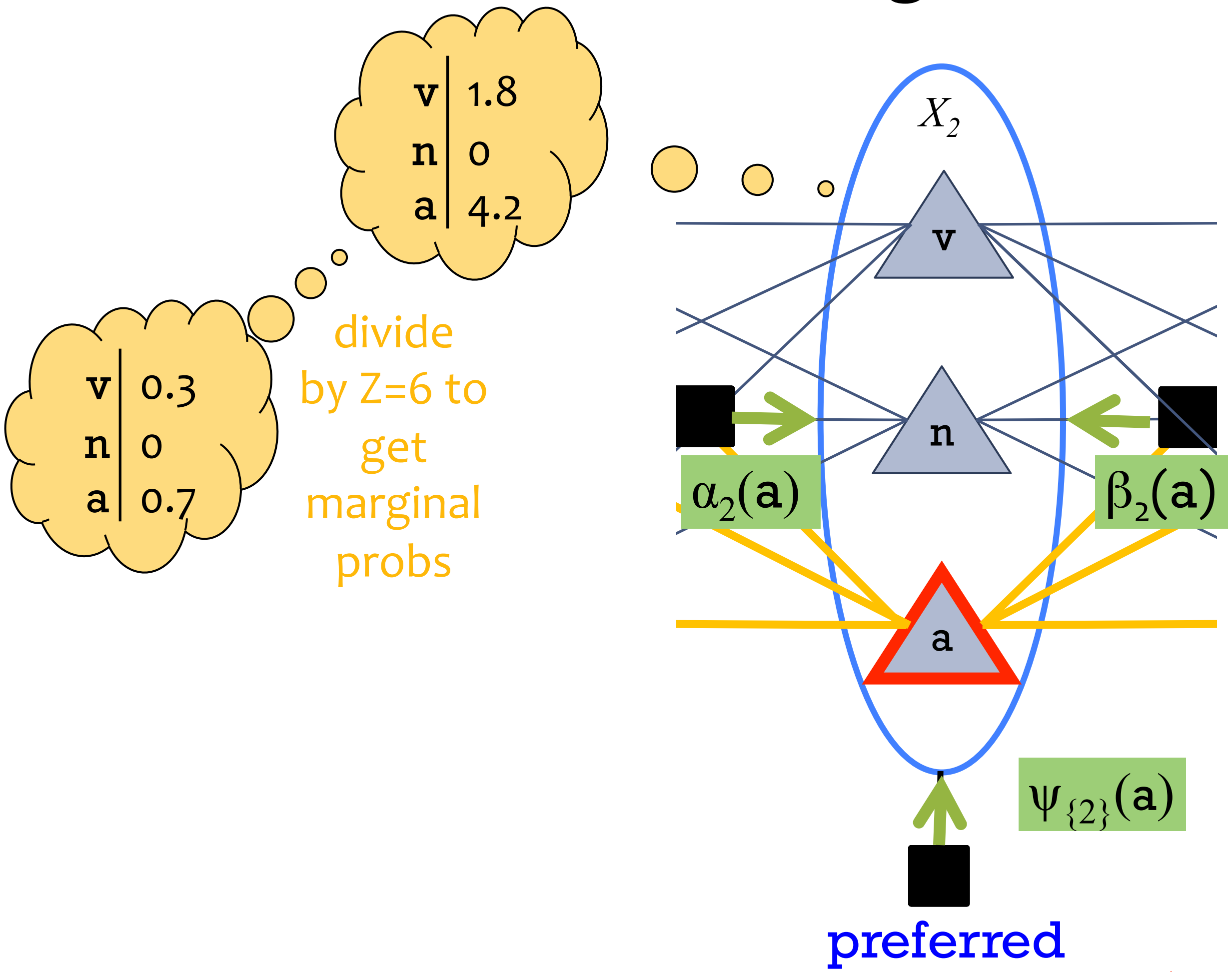
"belief that $X_2 = \mathbf{v}$"

"belief that $X_2 = \mathbf{n}$"

total weight of *all* paths through 

$= \alpha_2(\mathbf{v})\ \psi_{\{2\}}(\mathbf{v})\ \beta_2(\mathbf{v})$

Forward-Backward Algorithm: Finds Marginals



"belief that X_2 = **v**"

"belief that X_2 = **n**"

"belief that X_2 = **a**"

---

sum = Z
(total probability of *all* paths)

total weight of *all* paths through a

= $\alpha_2(\mathtt{a})$ $\psi_{\{2\}}(\mathtt{a})$ $\beta_2(\mathtt{a})$