First- and Second-order Expectation Semirings
with Applications to
Minimum-Risk Training on Translation Forests

Zhifei Li and Jason Eisner

Center for Language and Speech Processing

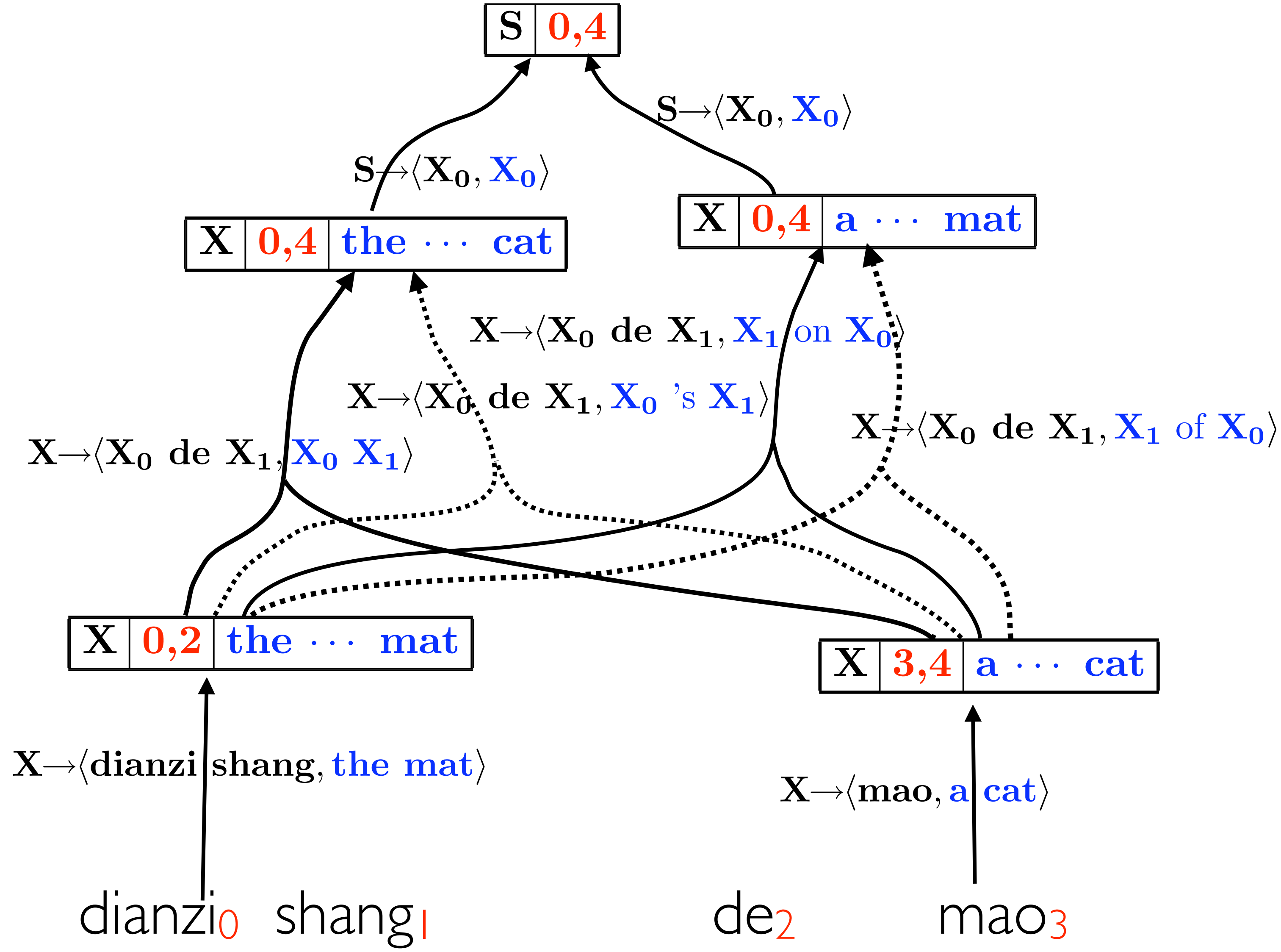
Computer Science Department

Johns Hopkins University

dianzi shang de mao

the cat on the mat

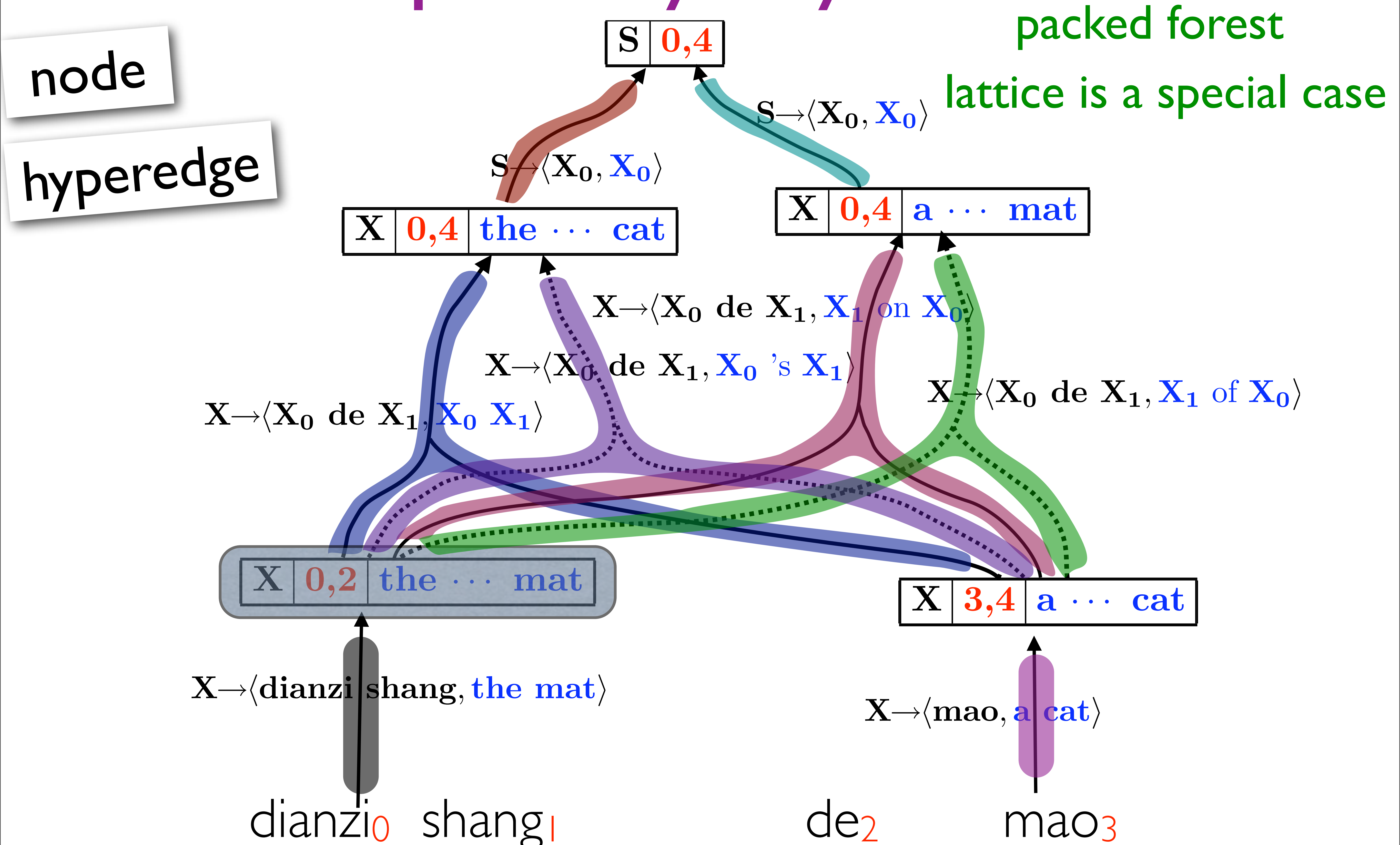
hypergraph



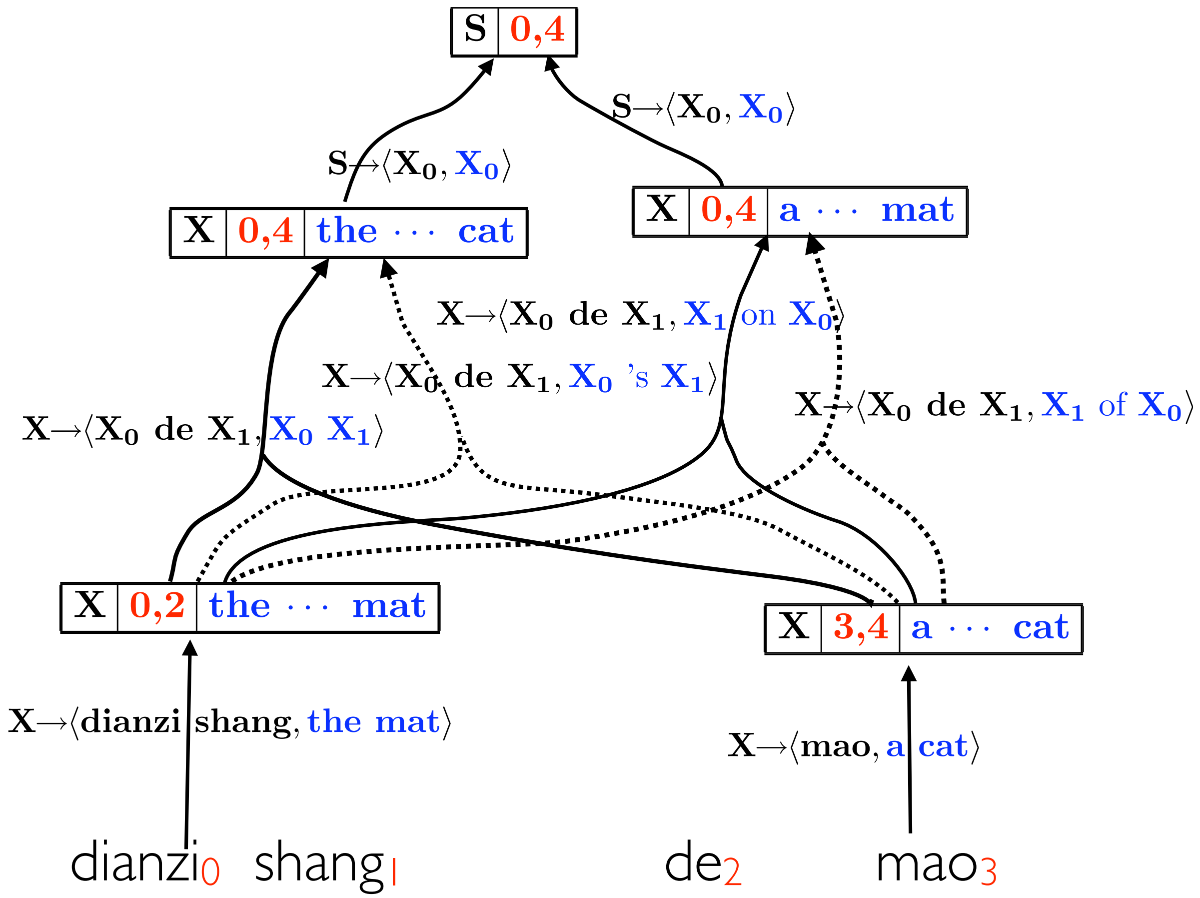
Joshua

dianzi shang de mao

A hypergraph is a compact data structure to encode exponentially many trees.



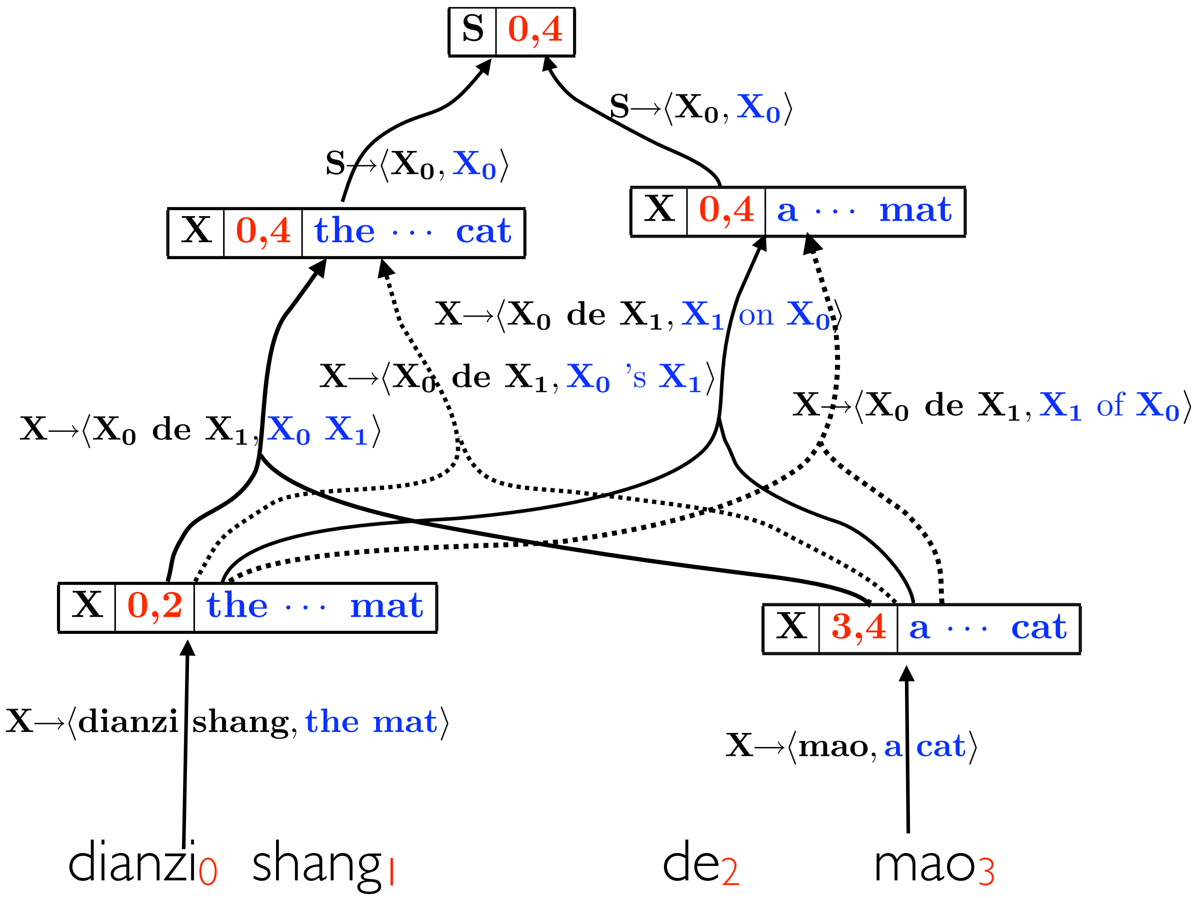
How many trees?



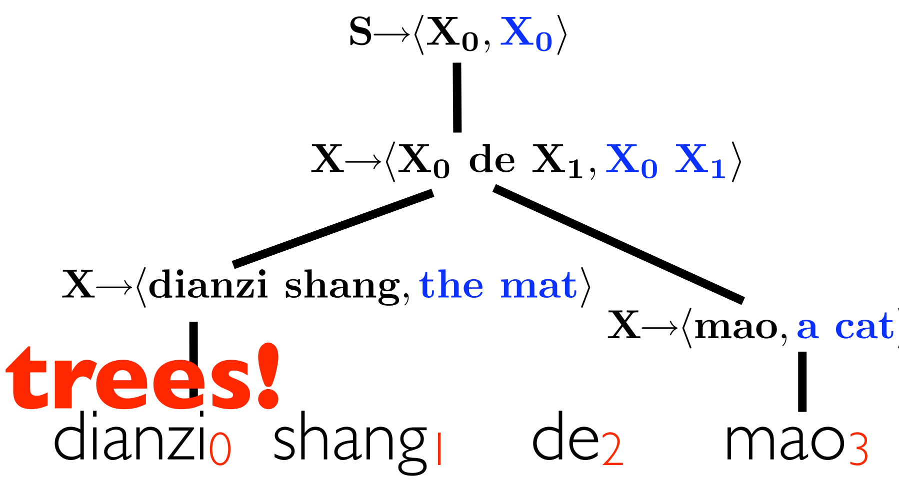
Monday, August 17, 2009

How many trees?

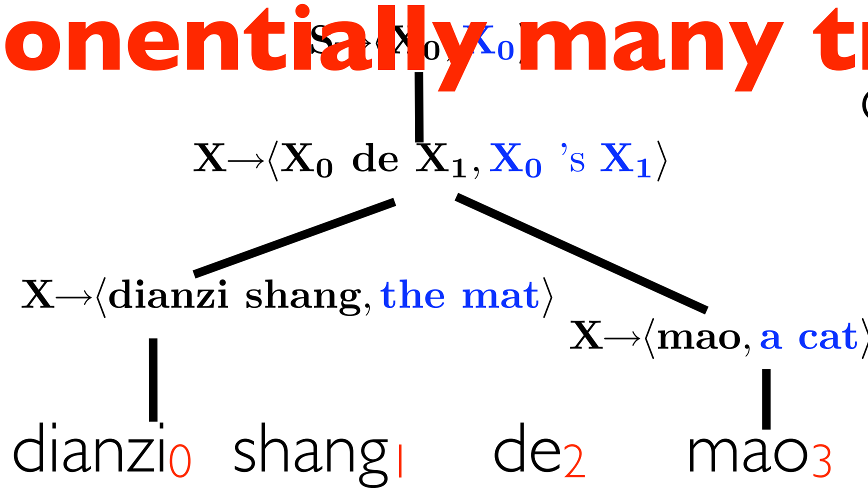
four ☺



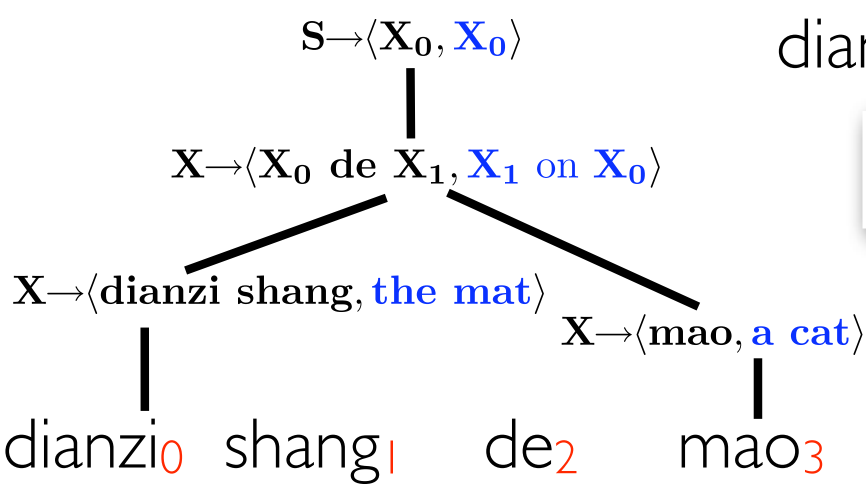
In general, exponentially many trees!



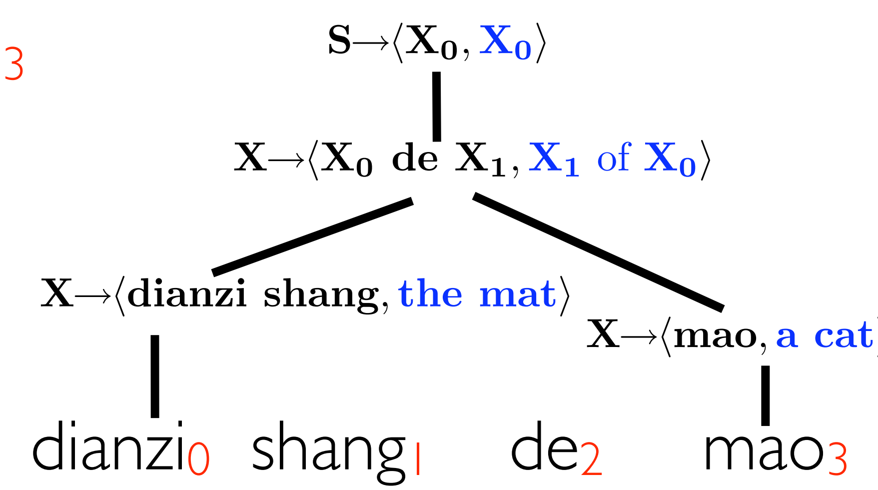
the mat a cat



the mat 's a cat

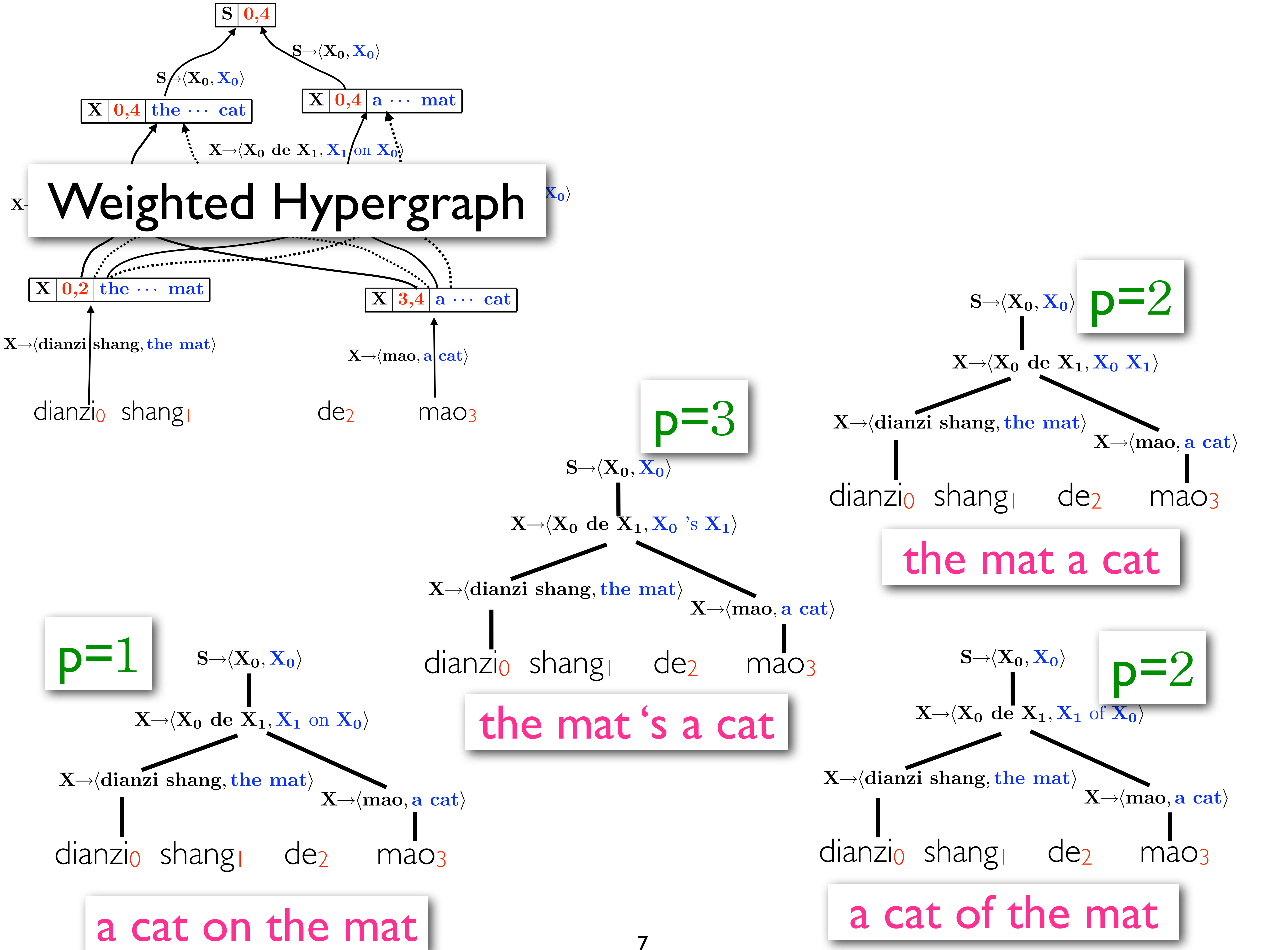


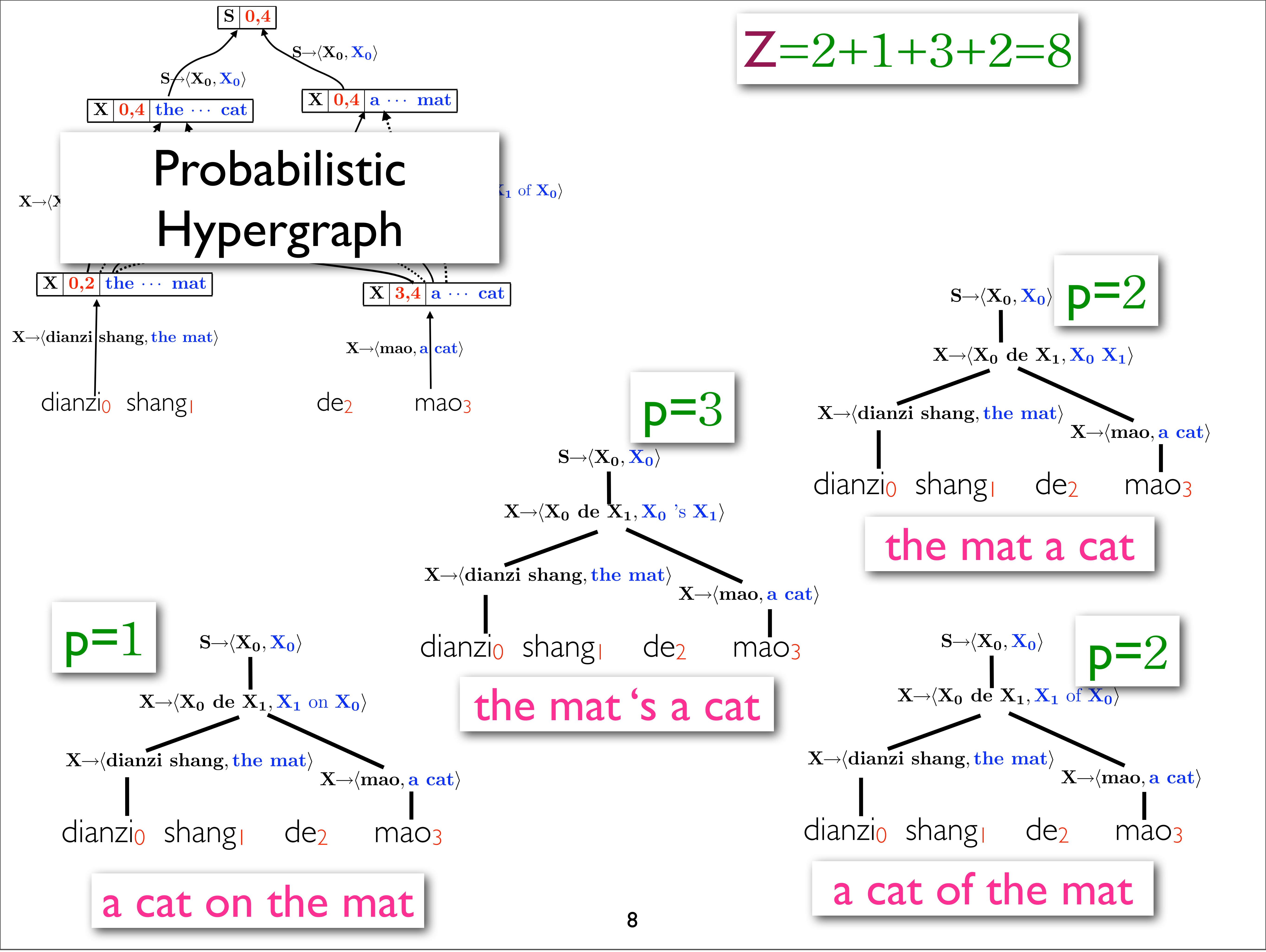
a cat on the mat



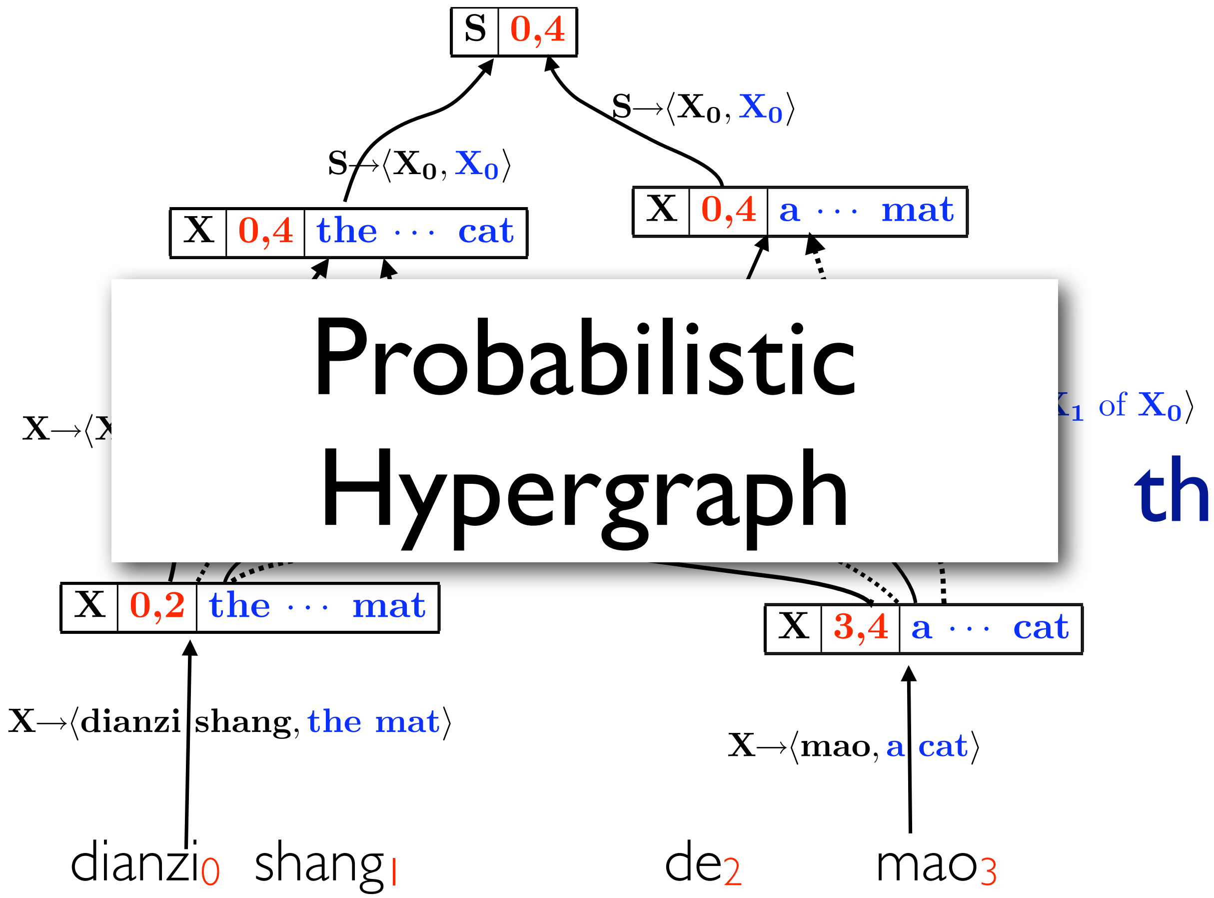
a cat of the mat

Weighted Hypergraph





Probabilistic Hypergraph
Z=2+1+3+2=8
S 0,4
S→⟨X0, X0⟩
X 0,4 the ··· cat
X 0,4 a ··· mat
X1 of X0⟩
X 0,2 the ··· mat
X 3,4 a ··· cat
X→⟨dianzi shang, the mat⟩
X→⟨mao, a cat⟩
dianzi0 shang1 de2 mao3
p=1
X→⟨X0 de X1, X1 on X0⟩
a cat on the mat
p=3
X→⟨X0 de X1, X0 's X1⟩
the mat 's a cat
p=2
X→⟨X0 de X1, X0 X1⟩
the mat a cat
p=2
X→⟨X0 de X1, X1 of X0⟩
a cat of the mat
8

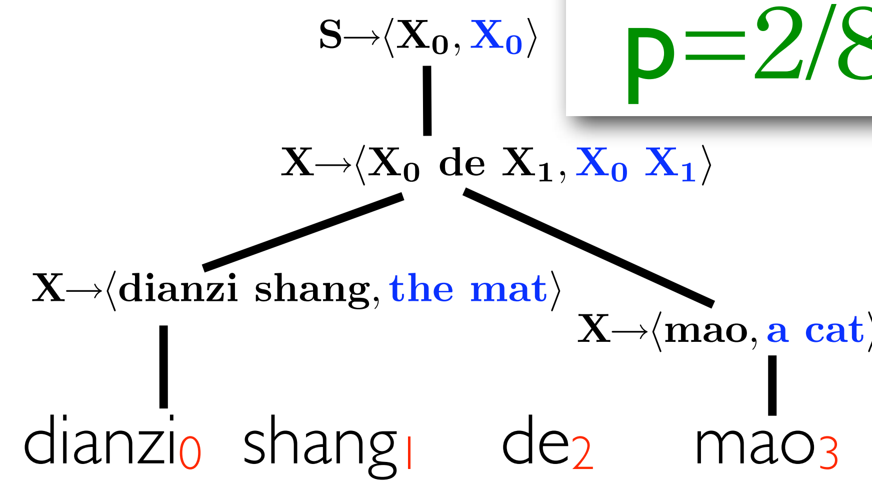


Probabilistic Hypergraph

Z=2+1+3+2=8

The hypergraph defines a probability distribution over **trees**!

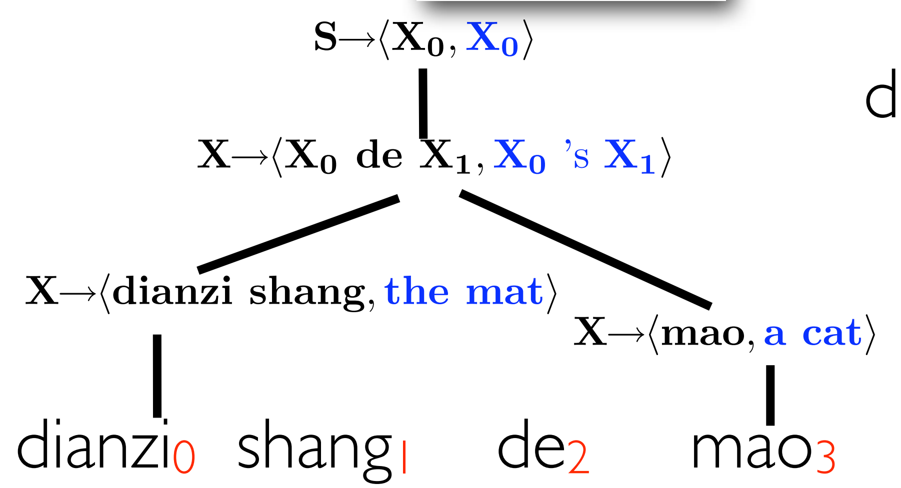
the distribution is parameterized by Θ



p=2/8

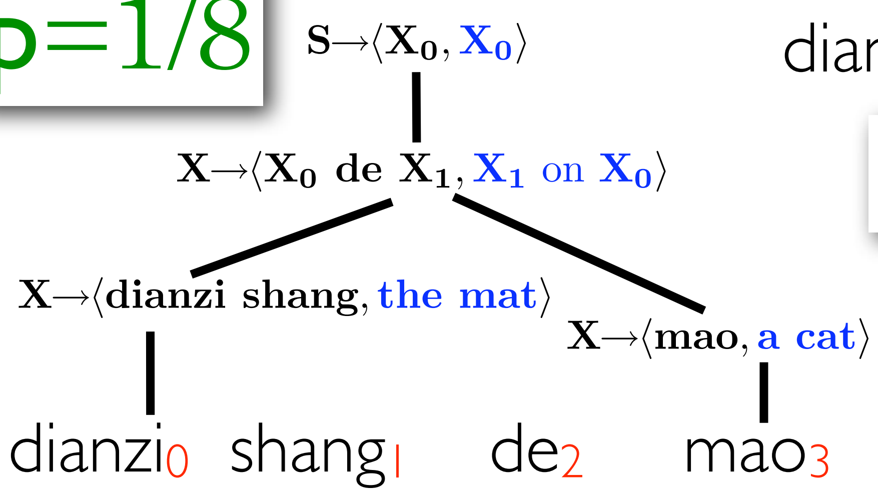
the mat a cat

p=3/8

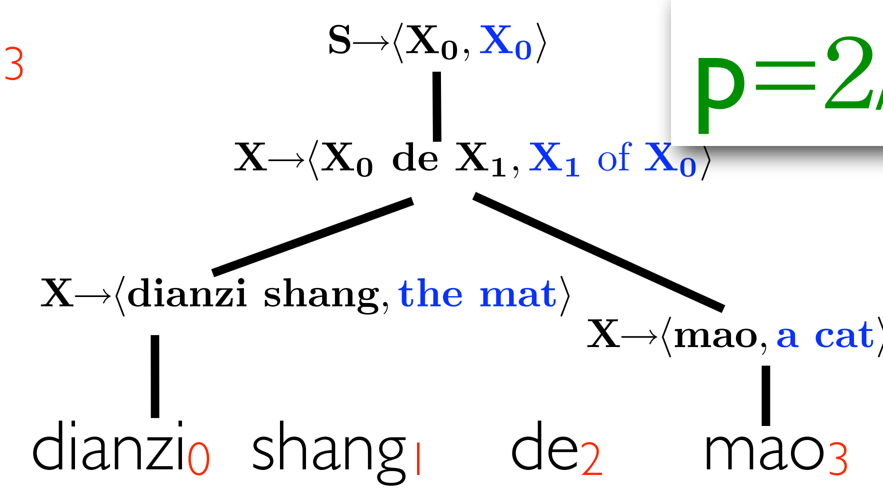


the mat 's a cat

p=1/8

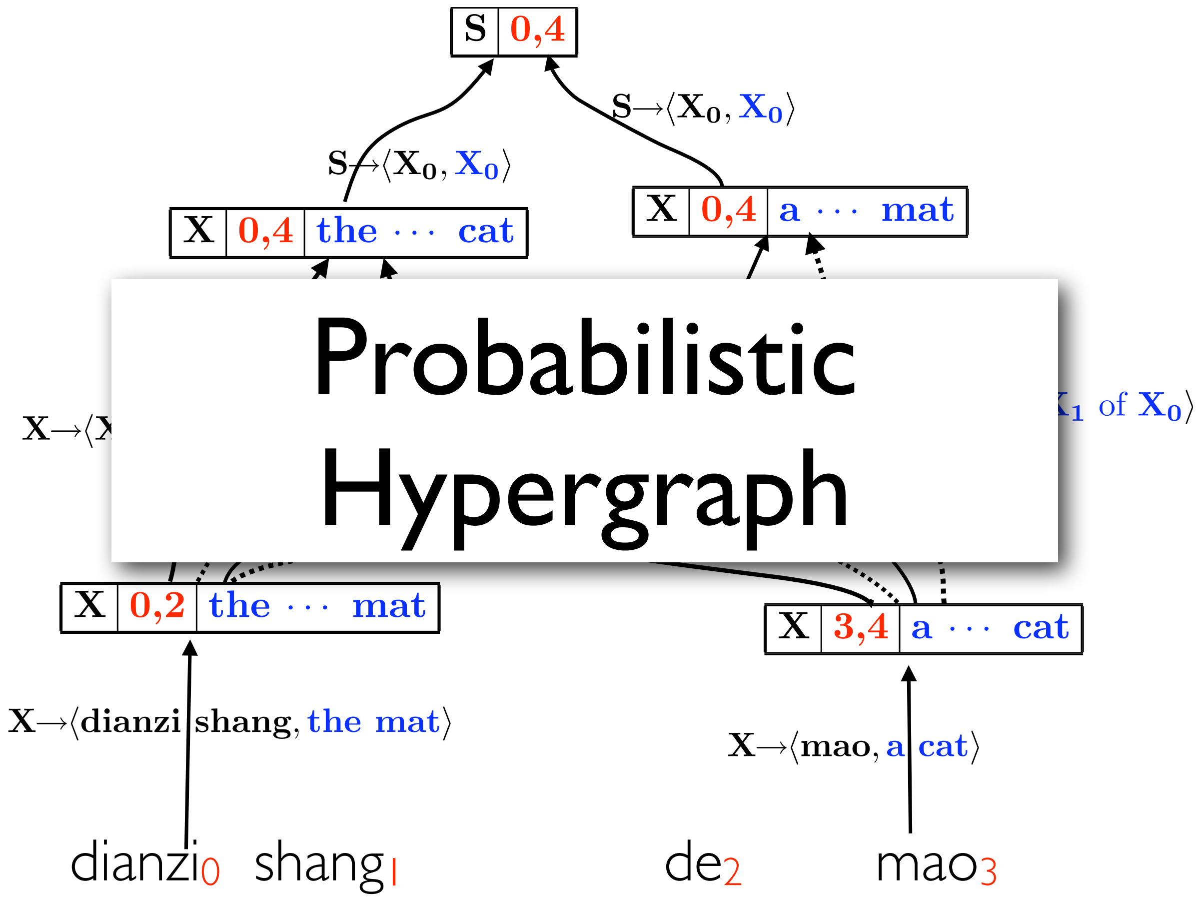


a cat on the mat



p=2/8

a cat of the mat



Probabilistic Hypergraph

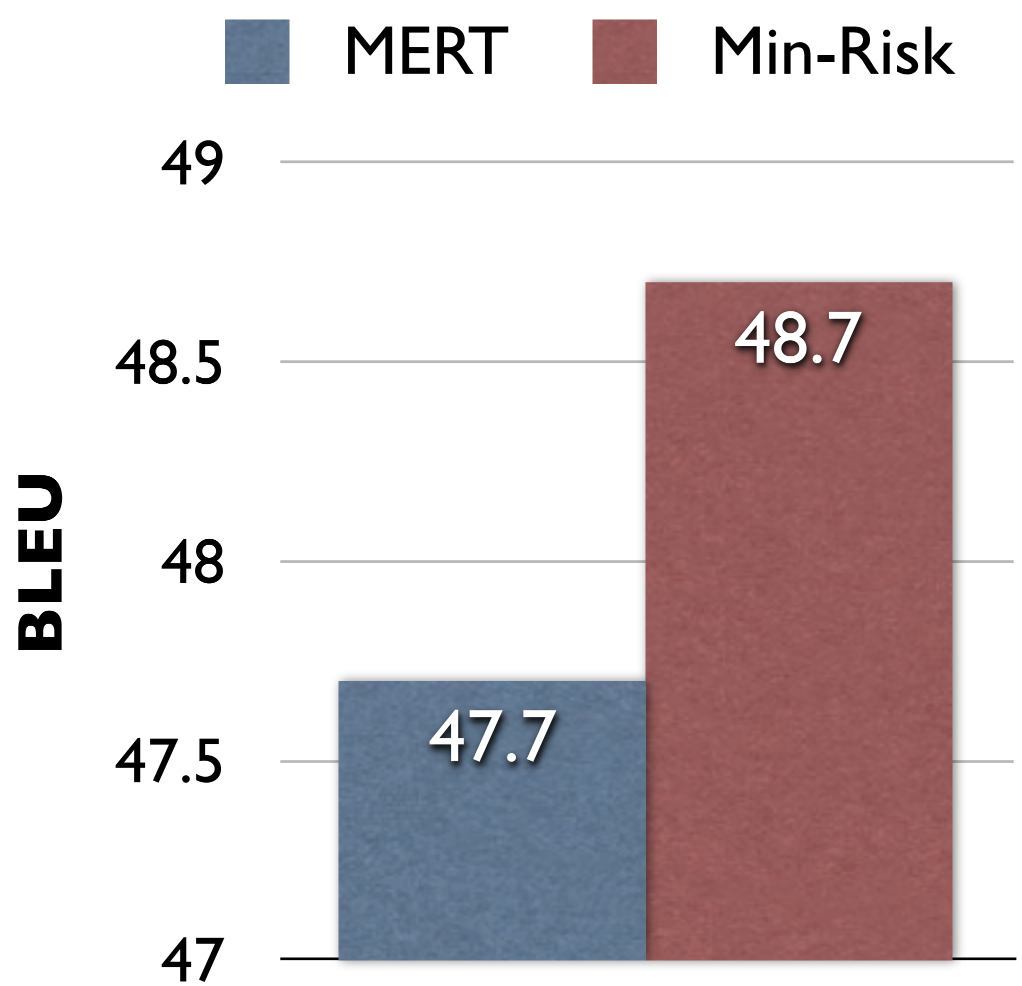
Entropy?

Bayes Risk?

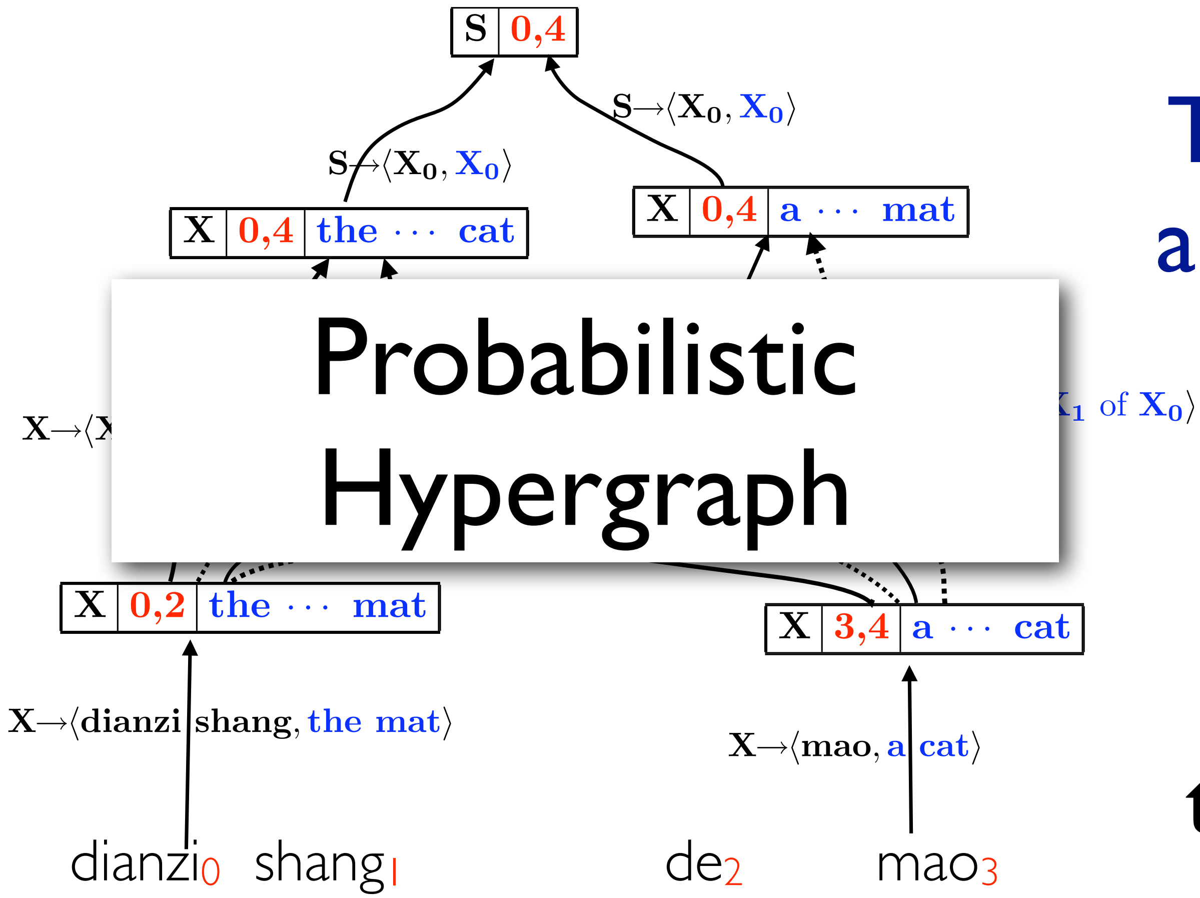
Minimum Risk Training on Translation Forests

Finding optimal parameters:

$$\theta^* = \mathit{argmin}\ \mathrm{Risk}(P_\theta) - \mathrm{Temperature} \times \mathrm{Entropy}(P_\theta)$$



BLEU scores on an IWSLT Chinese-English test set



This work provides a unified, elegant, and efficient framework in computing all of these!

useful in applying machine learning techniques into machine translation

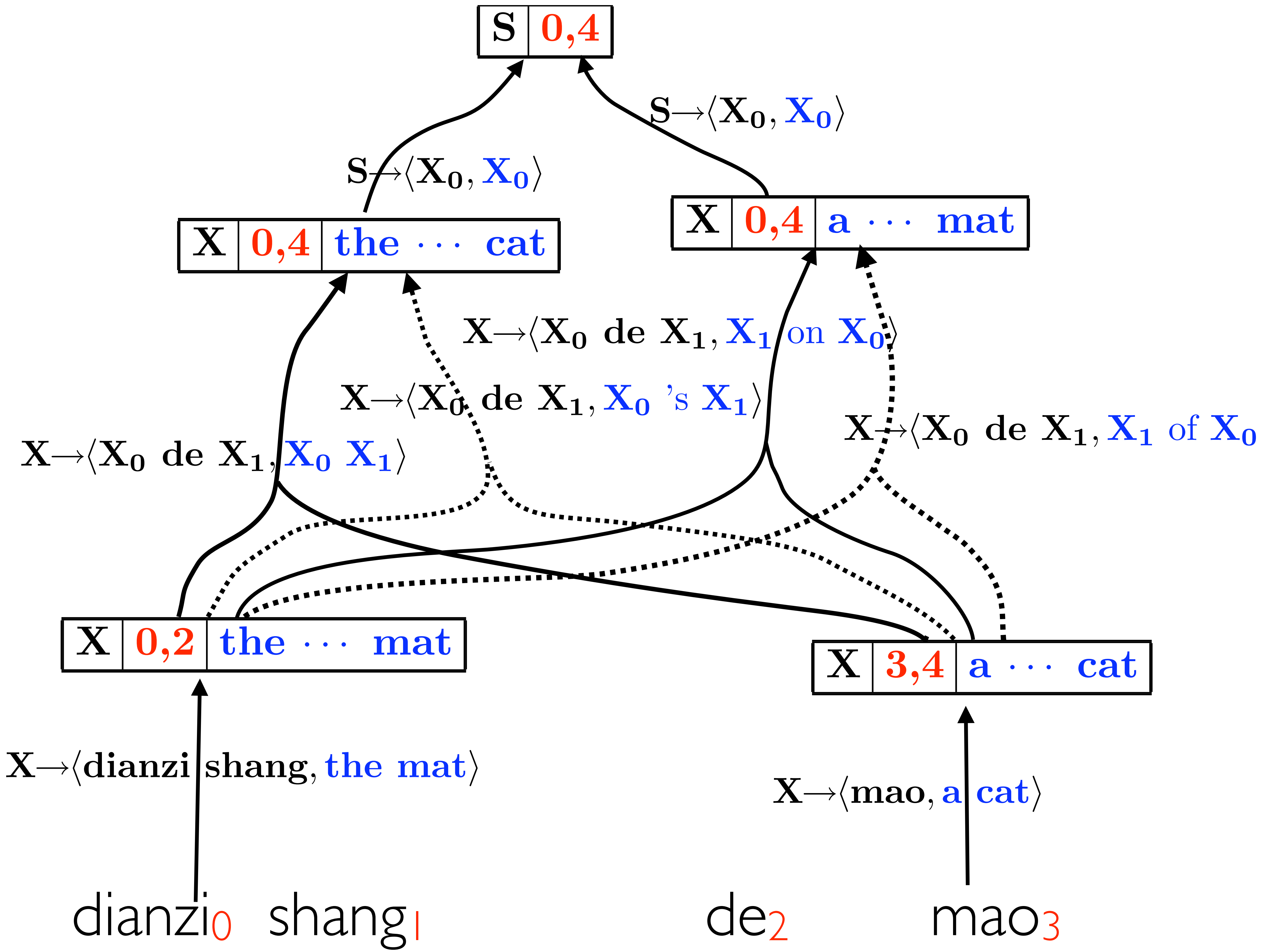
- First-order quantities:
  - expectation
    - entropy
    - Bayes risk
    - cross-entropy
    - KL divergence
    - feature expectations
  - first-order gradient of Z

- Second-order quantities:
  - Expectation over product
    - interaction between features
  - Hessian matrix of Z
    - second-order gradient descent
  - gradient of expectation
    - gradient of entropy or Bayes risk

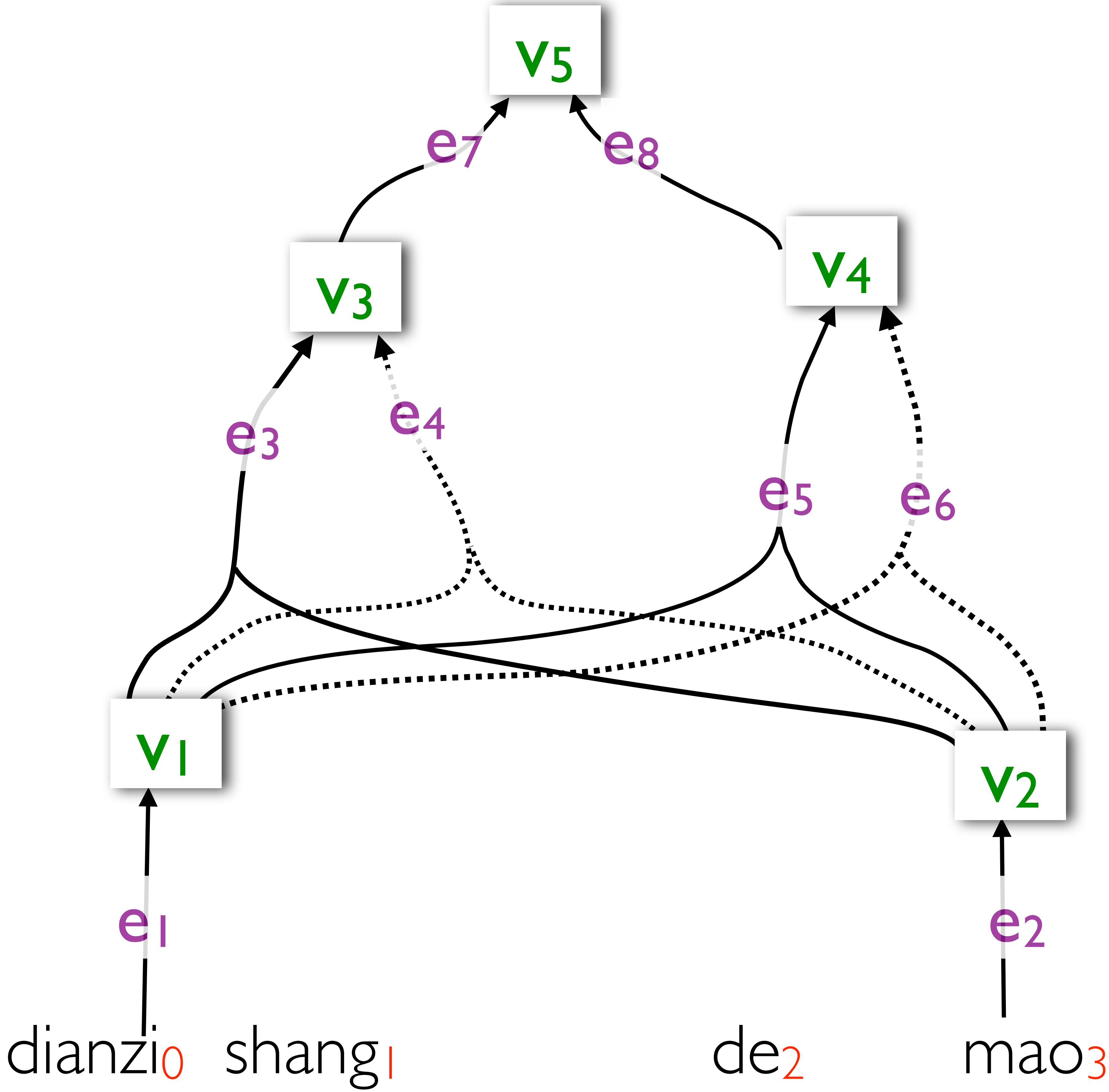
Outline

- Semiring-weighted Inside Algorithm
  - counting semiring (Goodman, 1999)
  - expectation semirings (Eisner, 2002)
  - second-order expectation semirings (new)
- Applications of the Semirings (new)
- Speed-up with Inside-outside (new)
- Minimum Risk Training over **Forests** (new)

Semiring-weighted Inside Algorithm



S | 0,4
S→⟨X0, X0⟩
S→⟨X0, X0⟩
X | 0,4 | the ··· cat
X | 0,4 | a ··· mat
X→⟨X0 de X1, X1 on X0⟩
X→⟨X0 de X1, X0 's X1⟩
X→⟨X0 de X1, X0 X1⟩
X→⟨X0 de X1, X1 of X0⟩
X | 0,2 | the ··· mat
X | 3,4 | a ··· cat
X→⟨dianzi shang, the mat⟩
X→⟨mao, a cat⟩
dianzi0 shang1 de2 mao3



v5
e7
e8
v3
v4
e3
e4
e5
e6
v1
v2
e1
e2
dianzi0 shang1 de2 mao3

What is a Semiring?

$$\langle K, \oplus, \otimes \rangle$$

a **set** with **plus** and **times** operations

Compute the Number of Derivation Trees

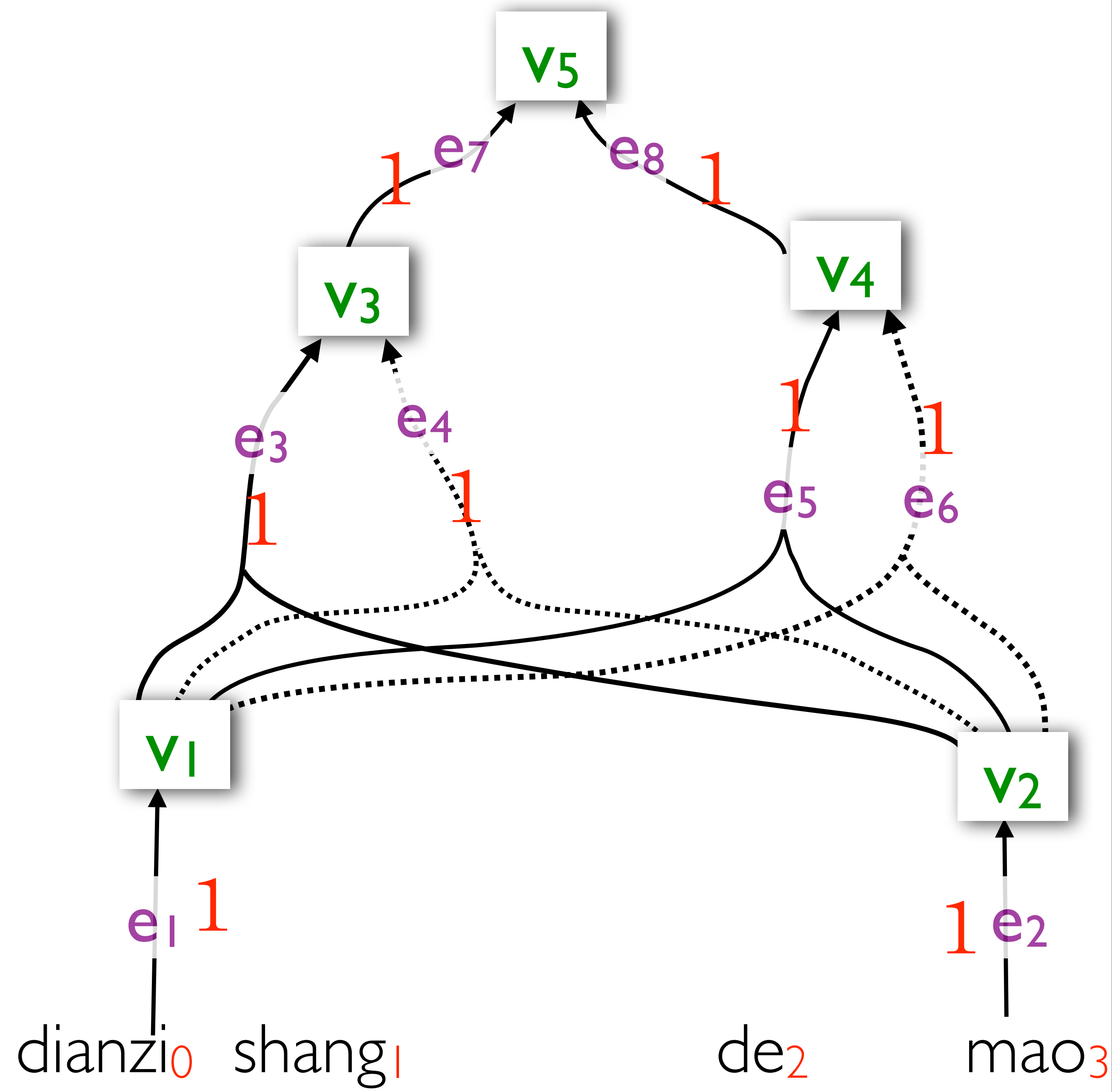
Counting semiring: ordinary integers

Inside algorithm:

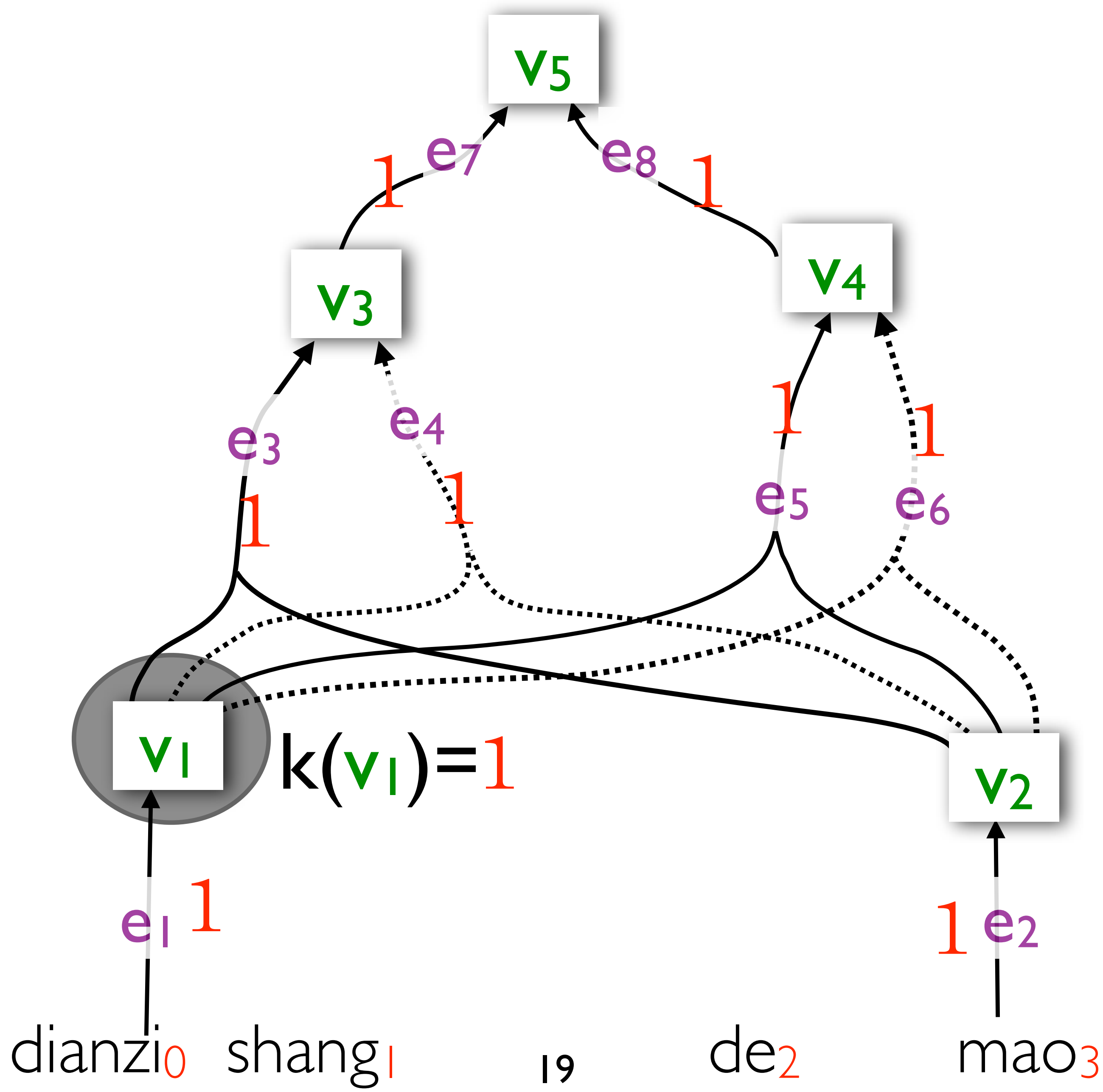
Inputs:
- a hypergraph
- a weight for each hyperedge

Output: k(root)

Complexity:
O(size of the hypergraph)



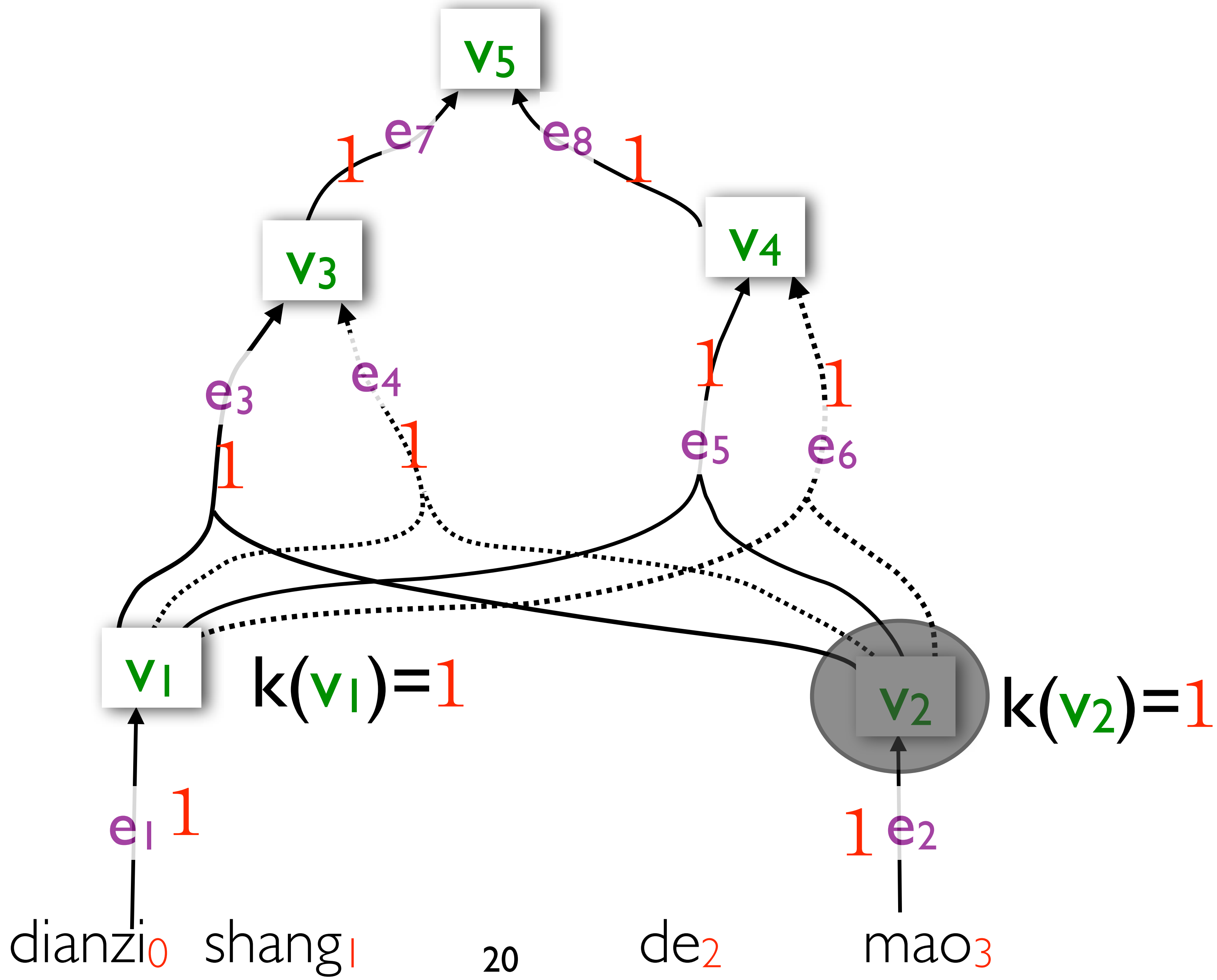
$k(v_1) = k(e_1)$



Bottom-up process in computing the number of trees

Monday, August 17, 2009

$k(v_1) = k(e_1)$ $k(v_2) = k(e_2)$

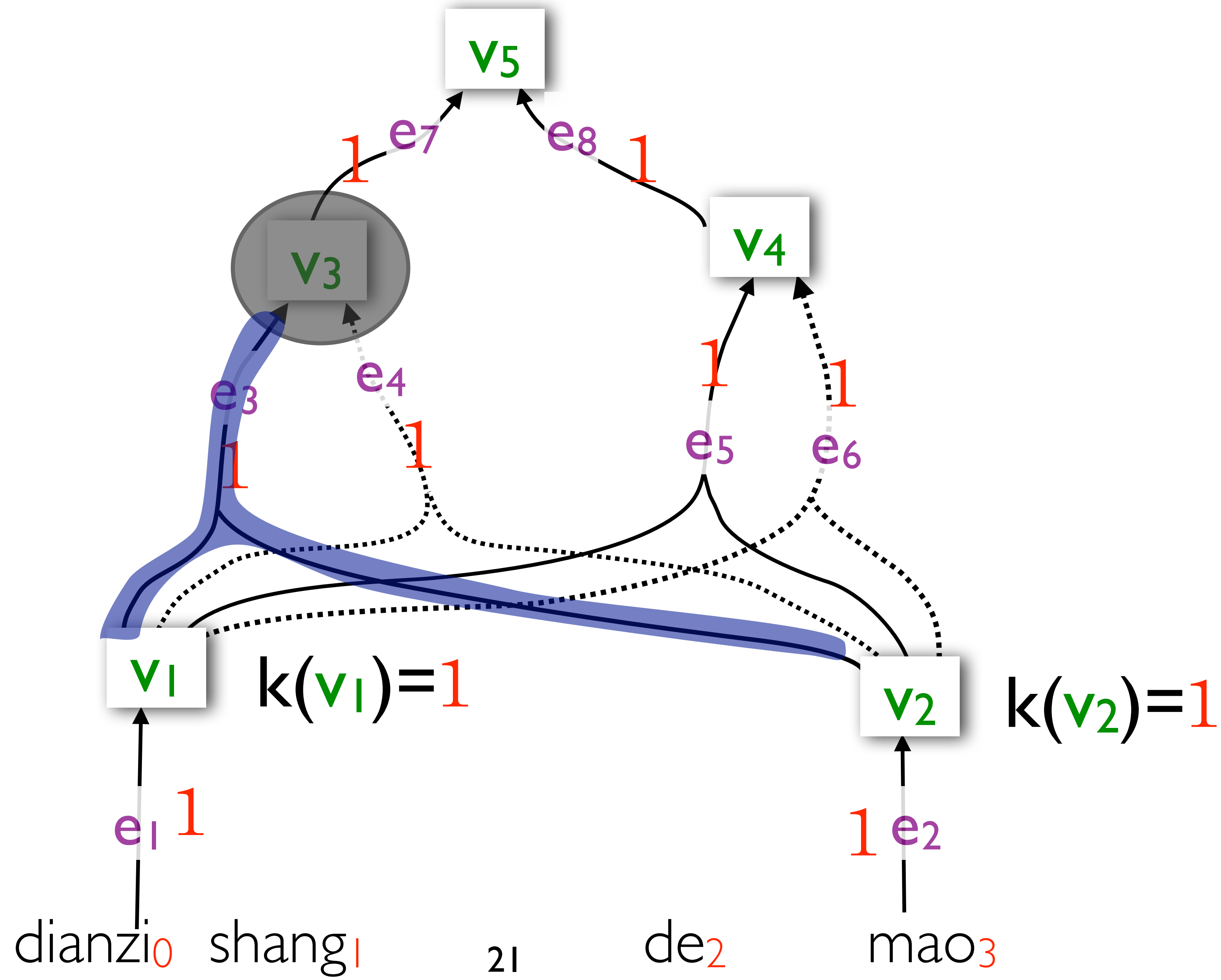


Bottom-up process in computing the number of trees

Compute $k(v_3)$: the weight at node v_3

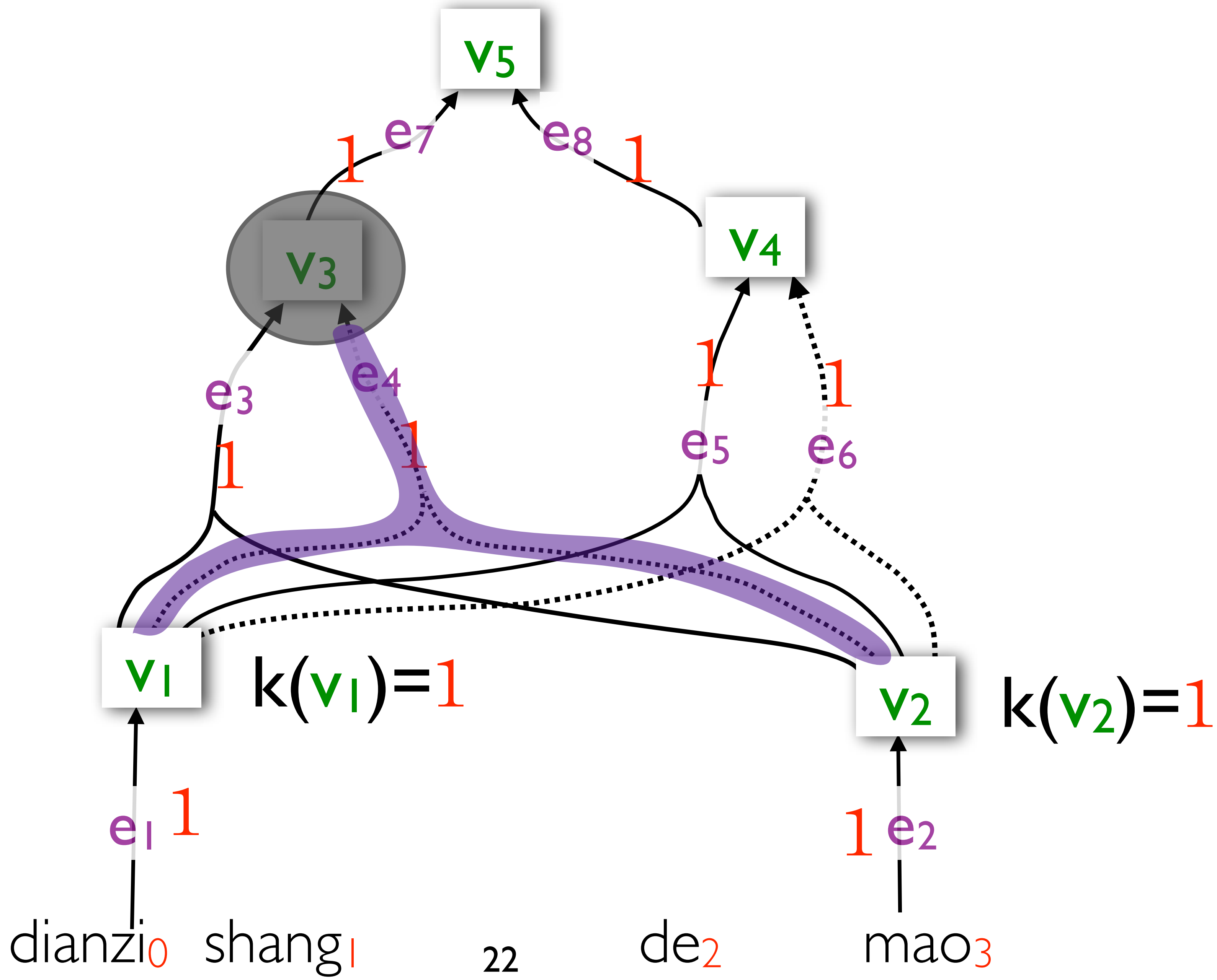
Hyperedge e_3: $k(e_3) \bigotimes k(v_1) \bigotimes k(v_2) = 1 \otimes 1 \otimes 1 = 1$

Bottom-up process in computing the number of trees



Compute $k(v_3)$: the weight at node v_3

Hyperedge e_4: $k(e_4) \bigotimes k(v_1) \bigotimes k(v_2) = 1 \otimes 1 \otimes 1 = 1$



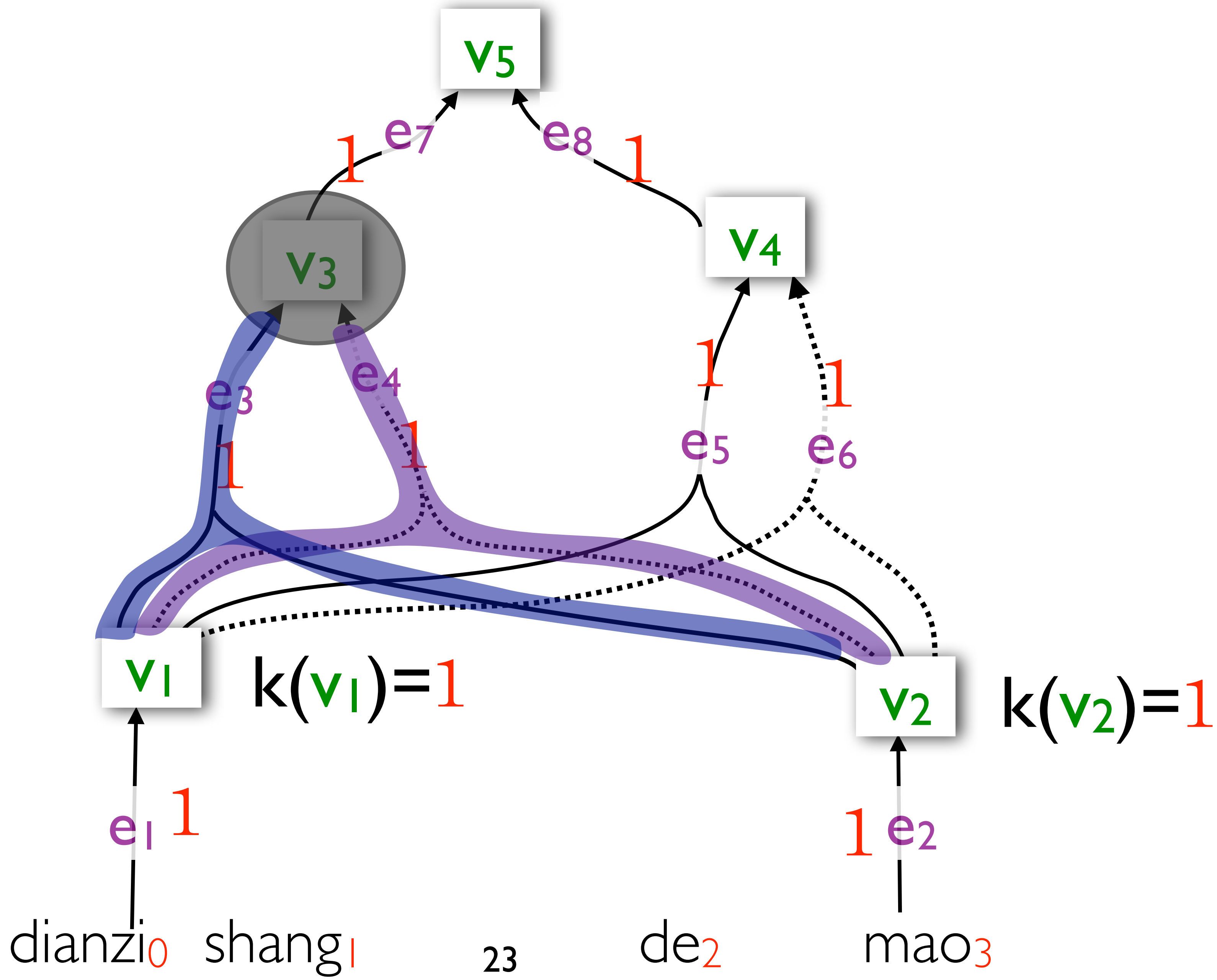
Bottom-up process in computing the number of trees

Compute $k(v_3)$: the weight at node v_3

Hyperedge e_3: $k(e_3) \otimes k(v_1) \otimes k(v_2) = 1$

Hyperedge e_4: $k(e_4) \otimes k(v_1) \otimes k(v_2) = 1$

Bottom-up process in computing the number of trees

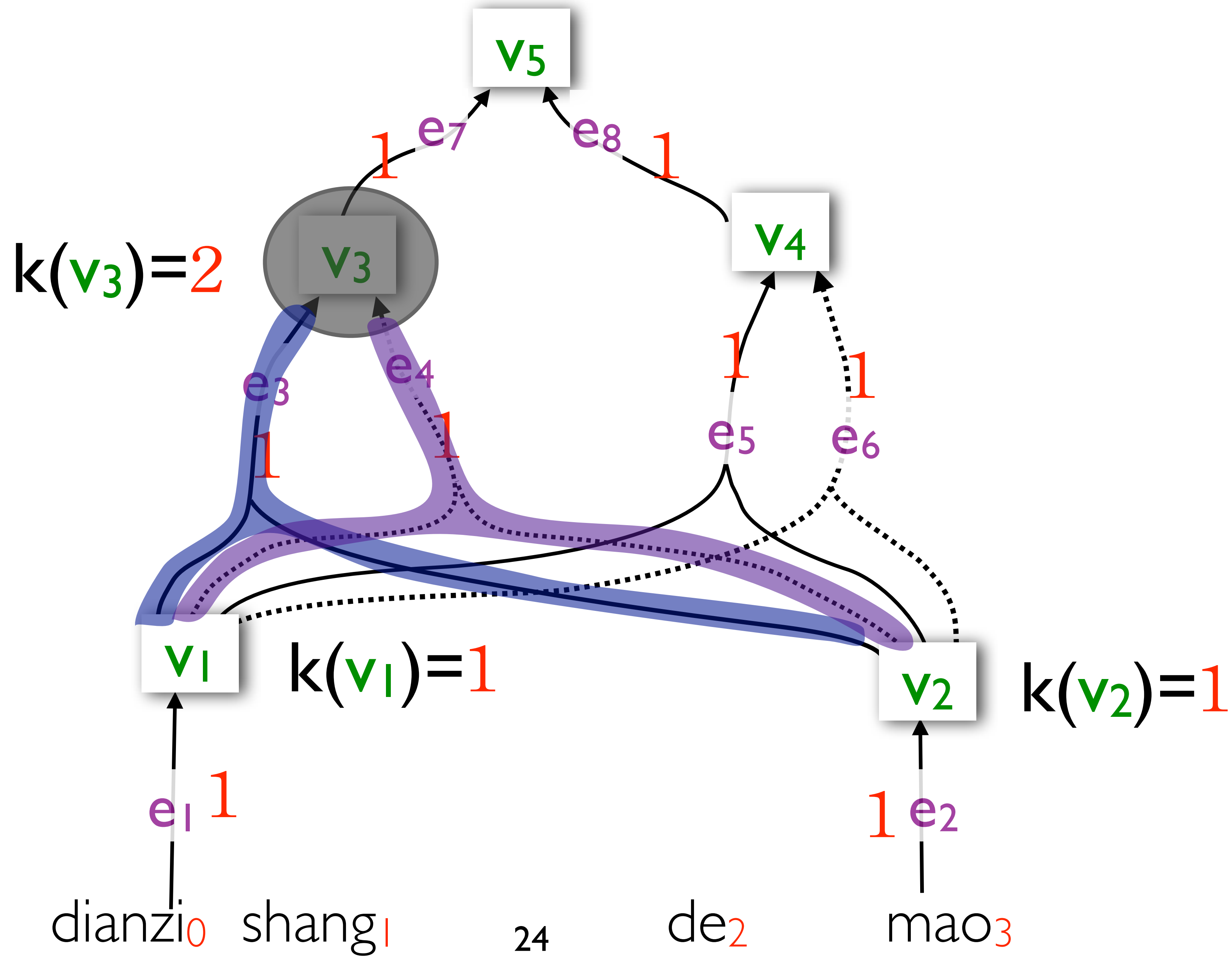


Compute $k(v_3)$: the weight at node v_3

$$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$$

$$1 \oplus 1 = 2$$

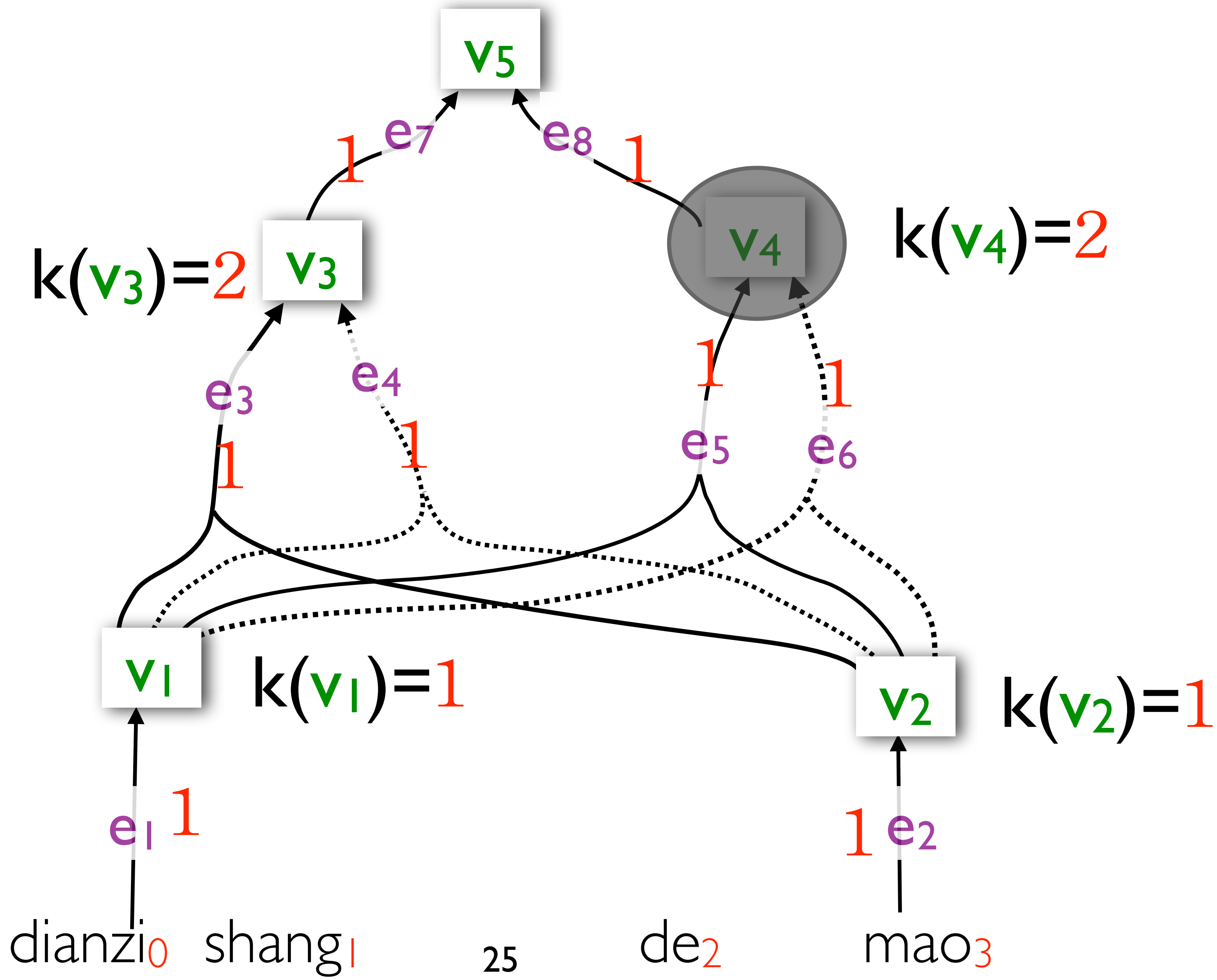
Bottom-up process in computing the number of trees



$k(v_1) = k(e_1)$ $\quad$ $k(v_2) = k(e_2)$

$$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$$

$$k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2)$$



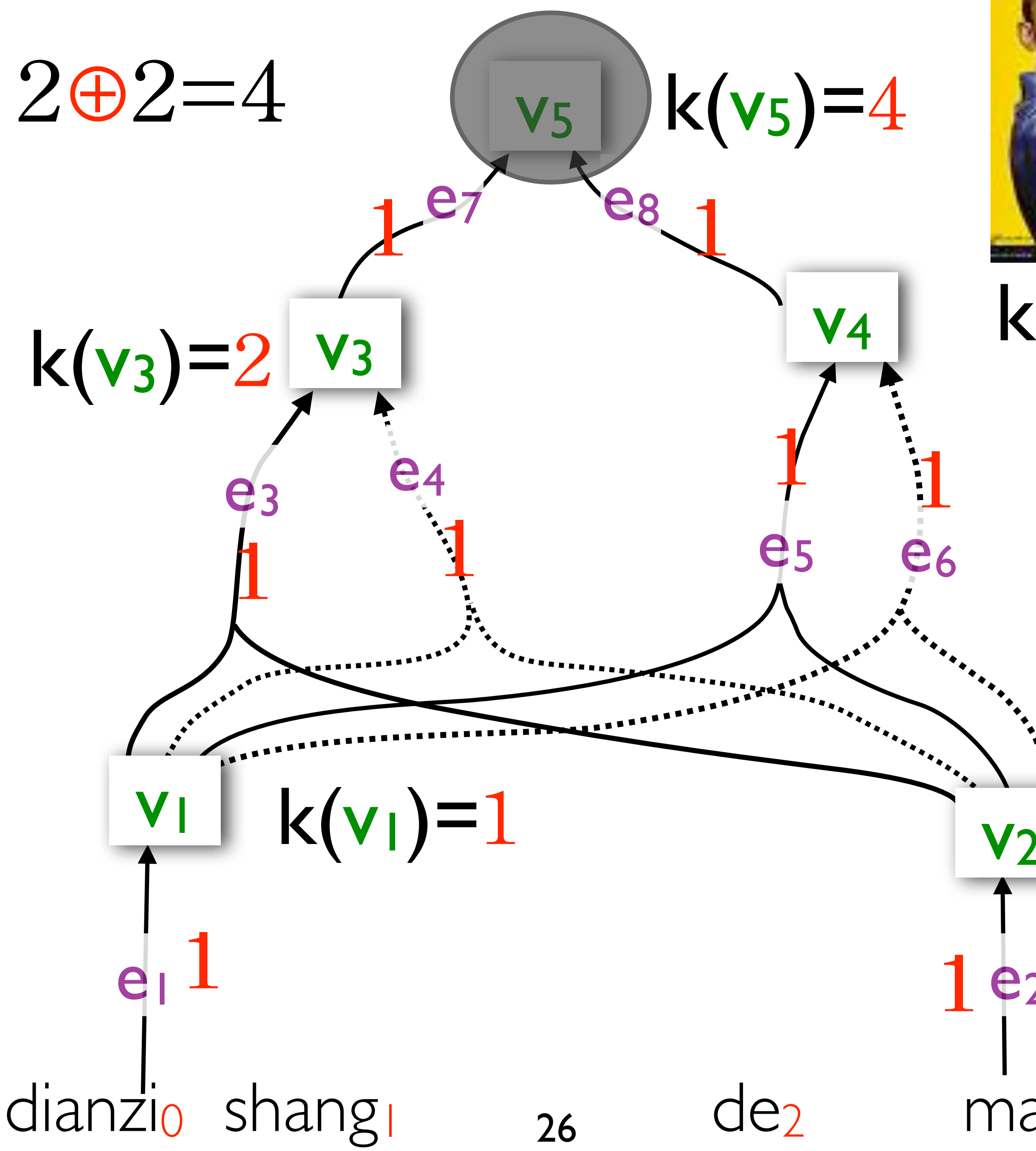
Bottom-up process in computing the number of trees

$k(v_1) = k(e_1)$ $\quad$ $k(v_2) = k(e_2)$

$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$

$k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2)$

$k(v_5) = k(e_7) \otimes k(v_3) \oplus k(e_8) \otimes k(v_4)$



Bottom-up

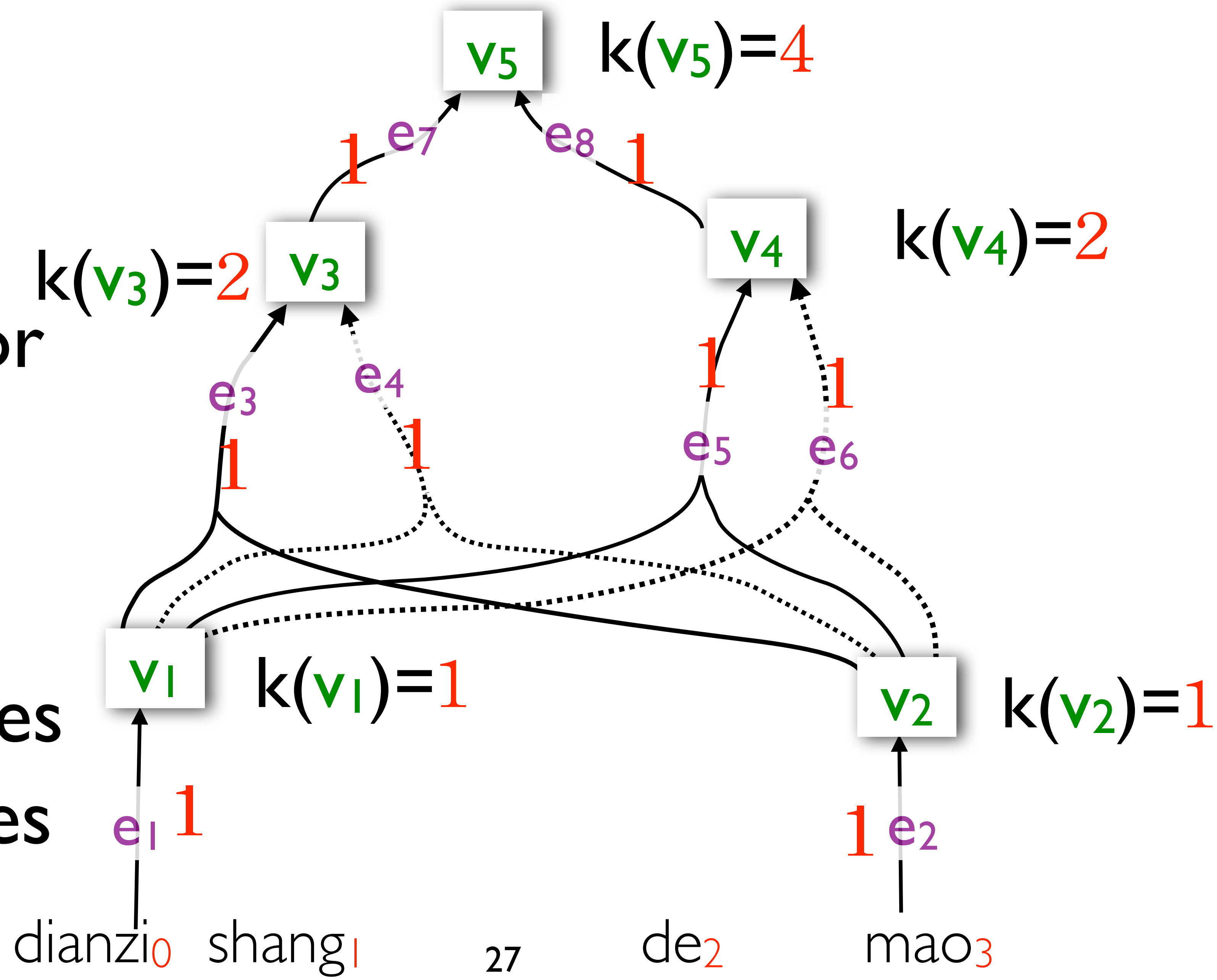
process in computing the number of trees

$k(v_1) = k(e_1)$ $\quad$ $k(v_2) = k(e_2)$

$$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$$

$$k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2)$$

$$k(v_5) = k(e_7) \otimes k(v_3) \oplus k(e_8) \otimes k(v_4)$$



Summary:

- **input**: a weight for each edge
- **output**: a weight for each node
- $\oplus$ is used at nodes
- $\otimes$ is used at edges

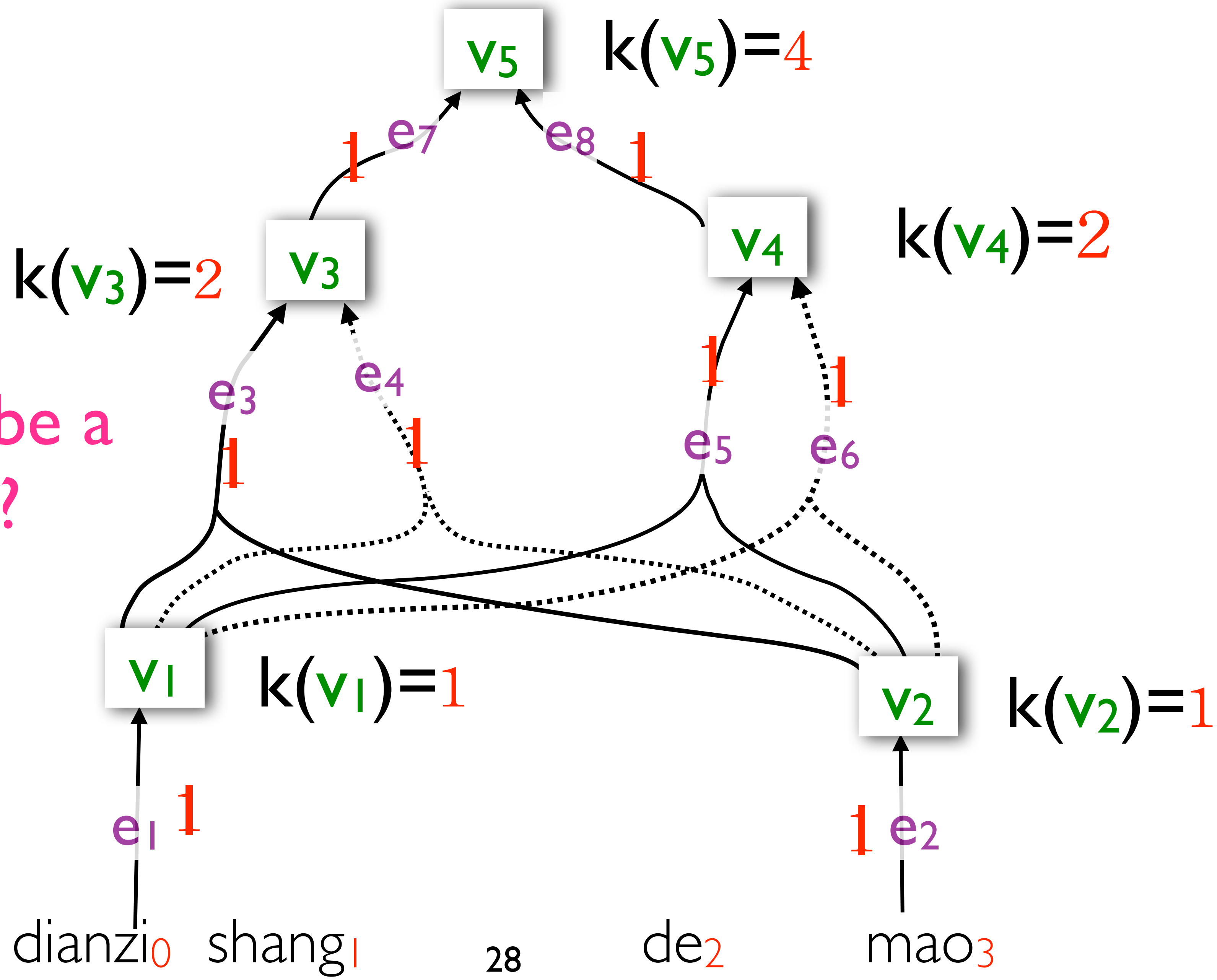
$k(v_1) = k(e_1)$ $\qquad$ $k(v_2) = k(e_2)$

$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$

$k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2)$

$k(v_5) = k(e_7) \otimes k(v_3) \oplus k(e_8) \otimes k(v_4)$

Does it have to be a single integer?



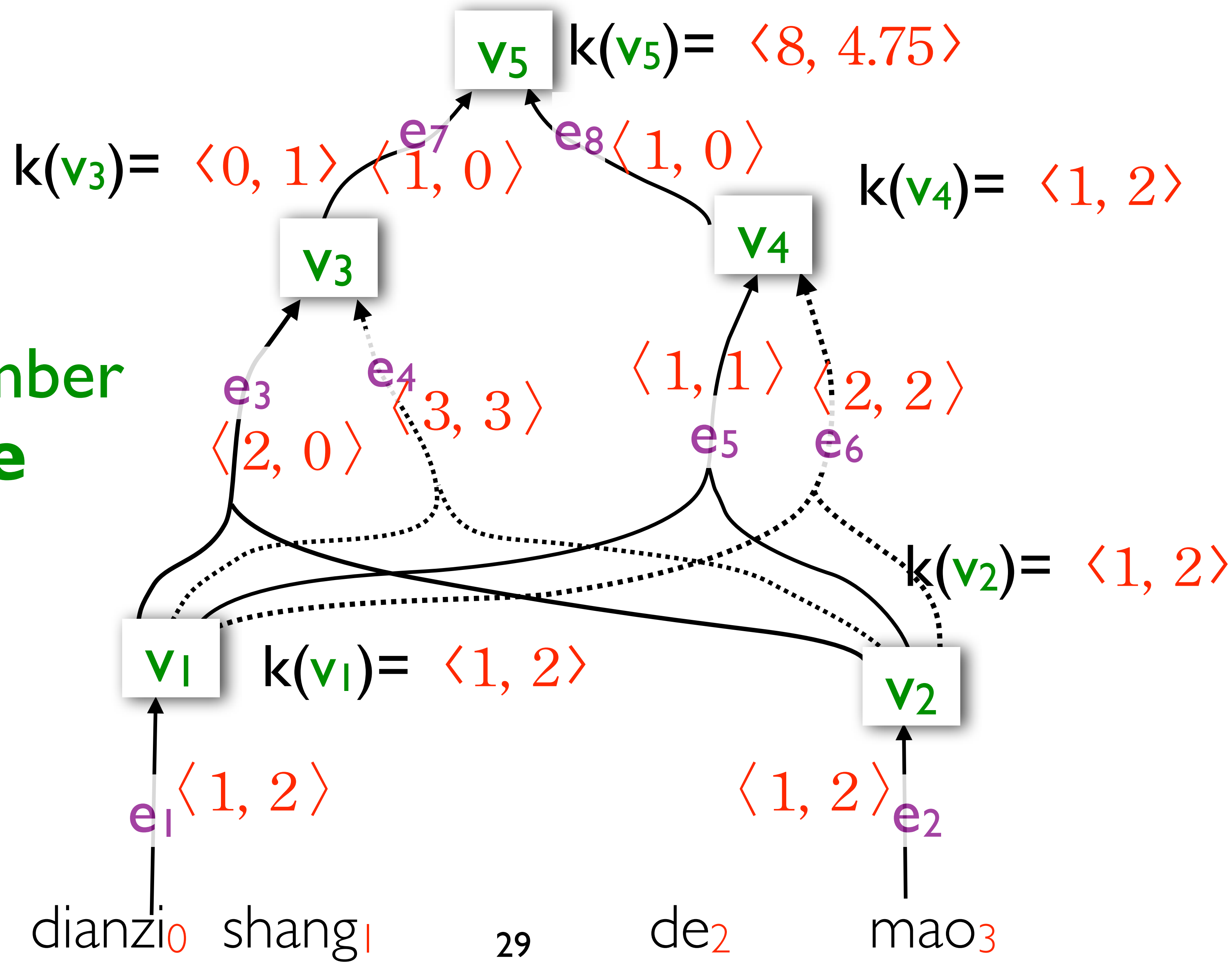
$$k(v_1) = k(e_1) \qquad k(v_2) = k(e_2)$$

$$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$$

$$k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2)$$

$$k(v_5) = k(e_7) \otimes k(v_3) \oplus k(e_8) \otimes k(v_4)$$

A semiring member is a **2-tuple**

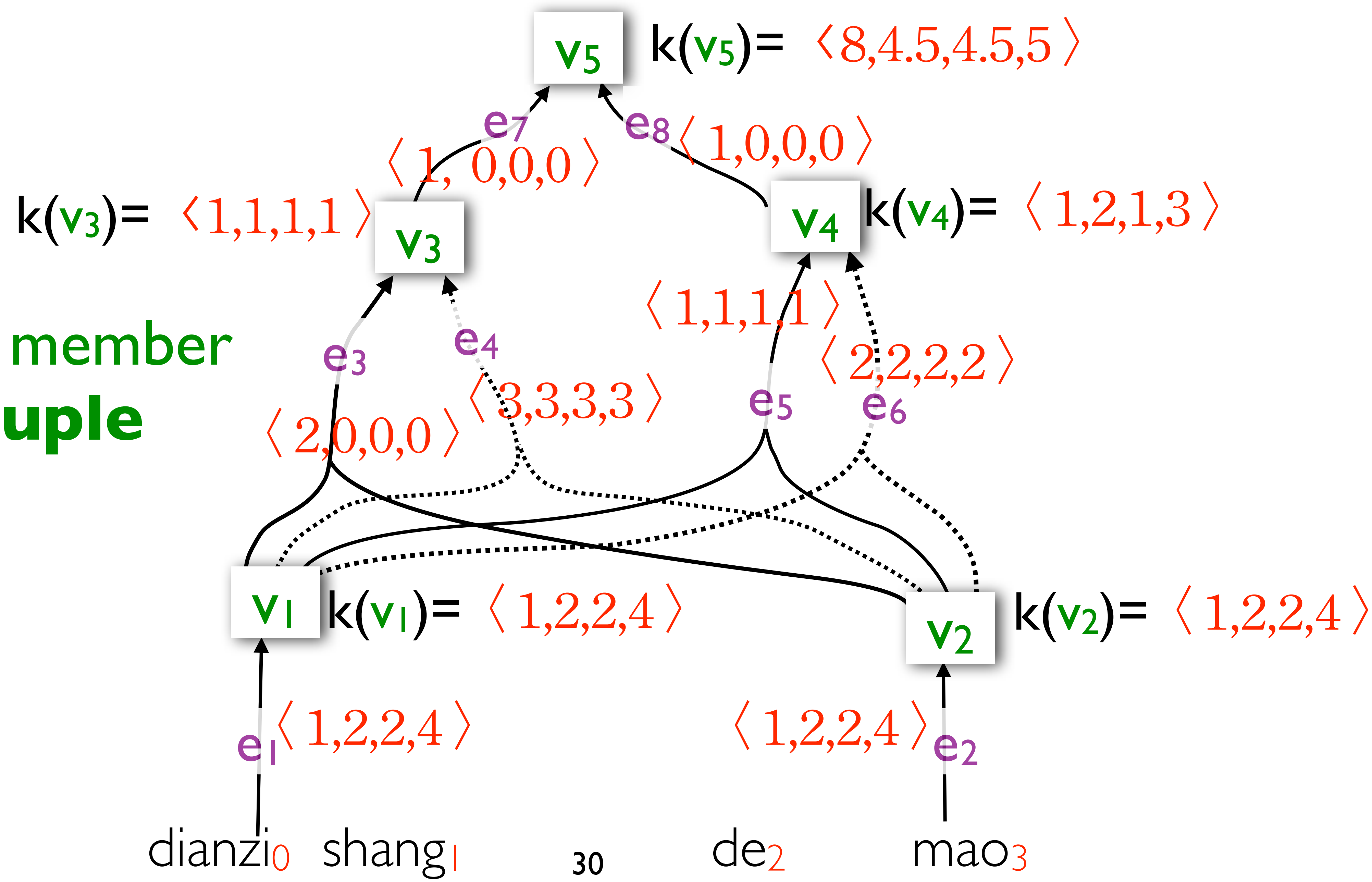


$$k(v_1) = k(e_1) \qquad\qquad k(v_2) = k(e_2)$$

$$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$$

$$k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2)$$

$$k(v_5) = k(e_7) \otimes k(v_3) \oplus k(e_8) \otimes k(v_4)$$



A semiring member is a 4-tuple

Why do we want to work on **tuples**?

expectation semiring

second-order expectation semiring

useful for
parameter estimation
at **training** time

Goodman (1999) defines many other semirings useful at "**testing**" time

First- and Second-order Expectation Semirings

First-order:

(Eisner, 2002)

- each member is a 2-tuple: $\langle p, r \rangle$

$\langle p_1, r_1 \rangle \otimes \langle p_2, r_2 \rangle$	$\langle p_1 p_2,\ p_1 r_2 + p_2 r_1 \rangle$
$\langle p_1, r_1 \rangle \oplus \langle p_2, r_2 \rangle$	$\langle p_1 + p_2,\ r_1 + r_2 \rangle$

Second-order:

- each member is a 4-tuple: $\langle p, r, s, t \rangle$

$\langle p_1, r_1, s_1, t_1 \rangle \otimes \langle p_2, r_2, s_2, t_2 \rangle$	$\langle p_1 p_2,\ p_1 r_2 + p_2 r_1,\ p_1 s_2 + p_2 s_1,$ $p_1 t_2 + p_2 t_1 + r_1 s_2 + r_2 s_1 \rangle$
$\langle p_1, r_1, s_1, t_1 \rangle \oplus \langle p_2, r_2, s_2, t_2 \rangle$	$\langle p_1 + p_2,\ r_1 + r_2,\ s_1 + s_2,\ t_1 + t_2 \rangle$

First- and Second-order Expectation Semirings

First-order:

- each member is a 2-tuple: $\langle p, r \rangle$

$\langle p_1, r_1 \rangle \otimes \langle p_2, r_2 \rangle$	$\langle p_1 p_2,\ p_1 r_2 + p_2 r_1 \rangle$
$\langle p_1, r_1 \rangle \oplus \langle p_2, r_2 \rangle$	$\langle p_1 + p_2,\ r_1 + r_2 \rangle$

What does p, r, s, or t mean?

Second-order:

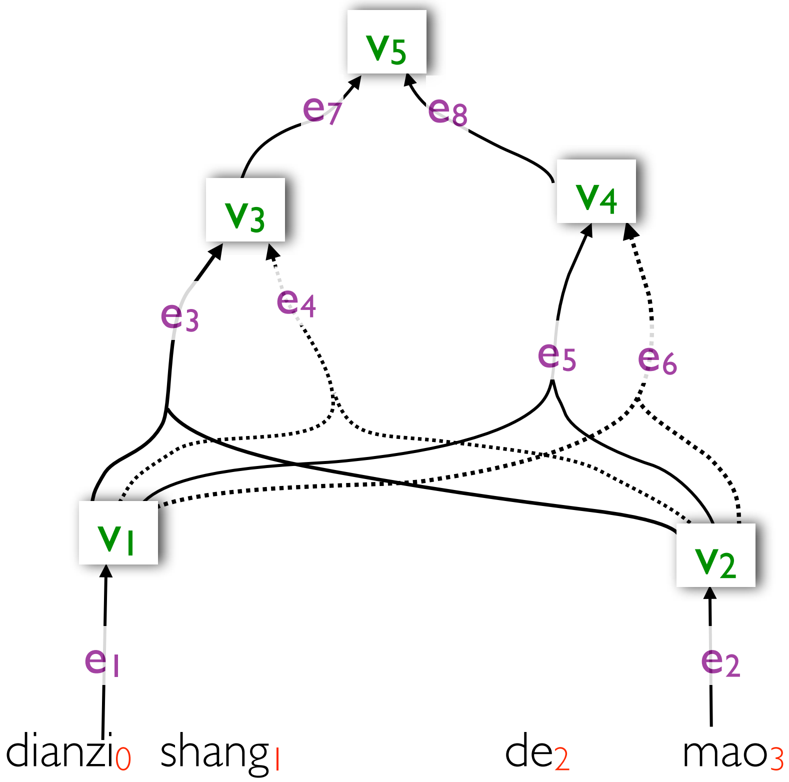
- ea

application-dependent

$\langle p_1, r_1, s_1, t_1 \rangle \otimes \langle p_2, r_2, s_2, t_2 \rangle$	$\langle p_1 p_2,\ p_1 r_2 + p_2 r_1,\ p_1 s_2 + p_2 s_1,$ $p_1 t_2 + p_2 t_1 + r_1 s_2 + r_2 s_1 \rangle$
$\langle p_1, r_1, s_1, t_1 \rangle \oplus \langle p_2, r_2, s_2, t_2 \rangle$	$\langle p_1 + p_2,\ r_1 + r_2,\ s_1 + s_2,\ t_1 + t_2 \rangle$

Applications

Compute Quantities on a Hypergraph: a Recipe



Three steps:

- choose a semiring

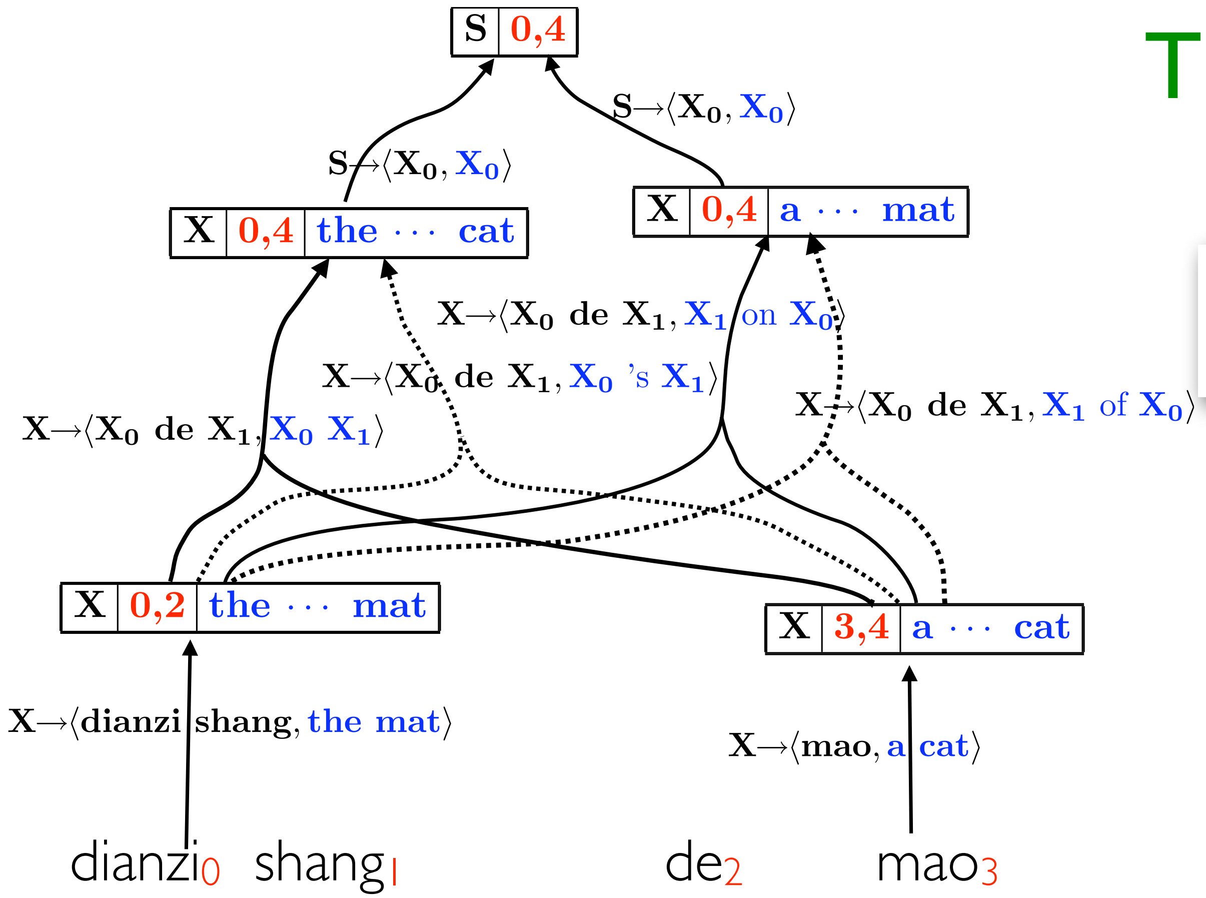
  first- or second-order expectation semiring

- specify a weight for each edge

- run the inside algorithm

Compute First-order Expectations

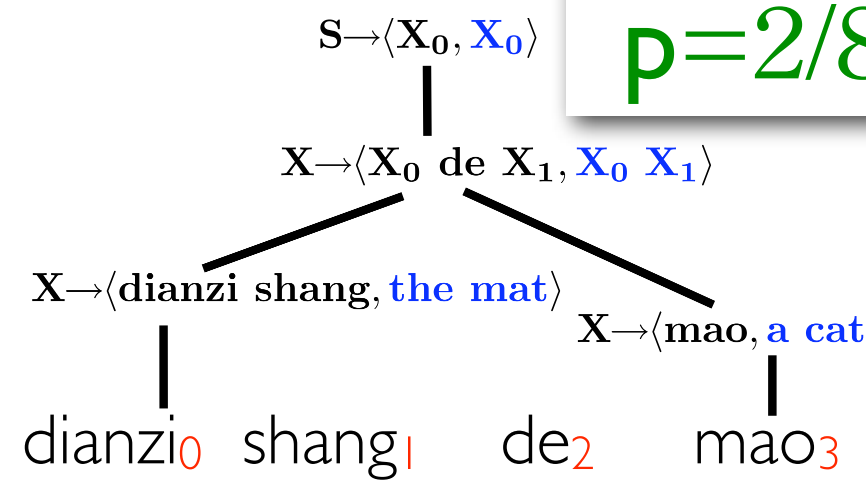
The hypergraph defines a probability distribution over trees!



expected translation length?

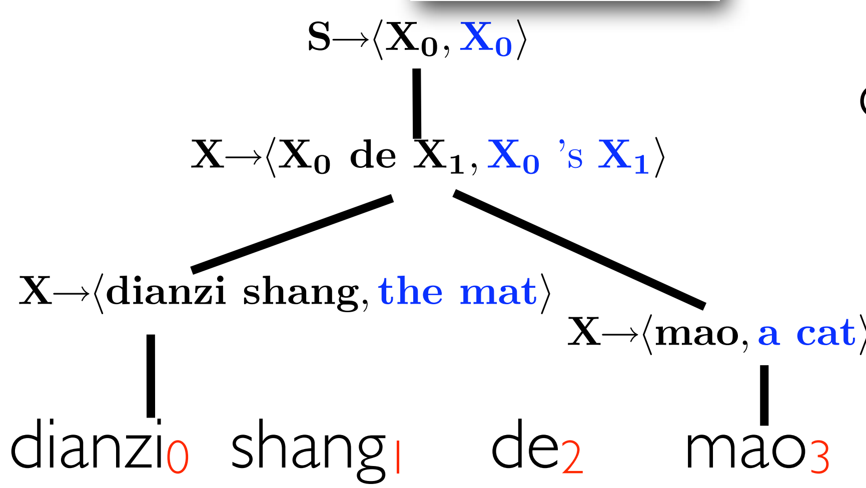
$4*2/8+5*6/8=4.75$

p=2/8



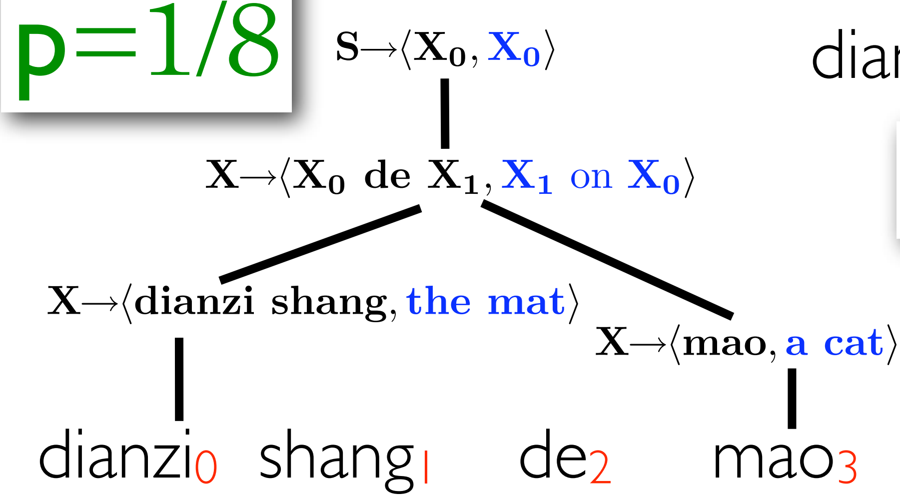
the mat a cat 4

p=3/8



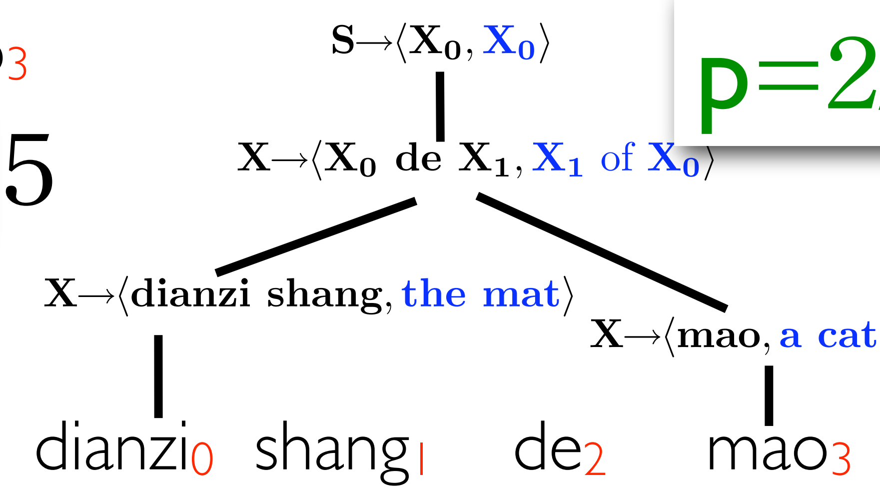
the mat 's a cat 5

p=1/8



a cat on the mat 5

p=2/8



a cat of the mat 5

Compute the expected translation length

using an expectation semiring

Compute Expected Translation Length

- choose a semiring

  expectation semiring

- specify a weight for each edge

$$k_e \stackrel{\text{def}}{=} \langle p_e, p_e r_e \rangle$$

p_e: transition probability or log-linear score at edge e

r_e: number of English words generated at edge e

- run the inside algorithm

Expectations on Hypergraphs

- Expectation over a hypergraph

$$\bar{r} \stackrel{\text{def}}{=} \mathbb{E}_p[r] = \sum_{d \in \text{HG}} p(d)r(d)$$

  - the distribution *p*(*d*) is defined by the hypergraph
  - *r*(*d*) is a function over a derivation *d*

    e.g., the length of the translation yielded by *d*

  - *r*(*d*) is additively decomposed

$$r(d) \stackrel{\text{def}}{=} \sum_{e \in d} r_e$$

e.g., translation length is additively decomposed!

Compute expectation using an expectation semiring:

$$k_e \stackrel{\text{def}}{=} \langle p_e, p_e r_e \rangle$$

p_e : transition probability or log-linear score at edge e

r_e ?

Entropy: $r_e \stackrel{\text{def}}{=} \log p_e$

Cross-entropy: $r_e \stackrel{\text{def}}{=} \log q_e$

Bayes risk: $r_e \stackrel{\text{def}}{=} \text{loss at edge } e$ (Tromble et al. 2008)

Why?

entropy is an **expectation**

$$\mathrm{H}(p) = \mathbb{E}_p[-\log p] = -\sum_{d \in \mathrm{HG}} p(d) \log p(d)$$

log *p*(*d*) is additively decomposed!

Compute expectation using an expectation semiring:

$$k_e \stackrel{\text{def}}{=} \langle p_e, p_e r_e \rangle$$

p_e : transition probability or log-linear score at edge e

r_e ?

Entropy: $r_e \stackrel{\text{def}}{=} \log p_e$

Cross-entropy: $r_e \stackrel{\text{def}}{=} \log q_e$

Bayes risk: $r_e \stackrel{\text{def}}{=} \text{loss at edge } e$ (Tromble et al. 2008)

Why?

cross-entropy is an **expectation**

$$\mathrm{H}(p, q) = \mathbb{E}_p(-\log q) = -\sum_{d \in \mathrm{HG}} p(d) \log q(d)$$

log $q(d)$ is additively decomposed!

Compute expectation using an expectation semiring:

$$k_e \stackrel{\text{def}}{=} \langle p_e, p_e r_e \rangle$$

p_e : transition probability or log-linear score at edge e

r_e ?

Entropy:

$$r_e \stackrel{\text{def}}{=} \log p_e$$

Cross-entropy:

$$r_e \stackrel{\text{def}}{=} \log q_e$$

Bayes risk:

$$r_e \stackrel{\text{def}}{=} \text{loss at edge } e$$

(Tromble et al. 2008)

Why?

Bayes risk is an **expectation**

$$\mathrm{H}(p,q) = \mathbb{E}_p(L) = -\sum_{d \in \mathrm{HG}} p(d) \cdot L(Y(d))$$

L(*Y*(*d*)) is additively decomposed!

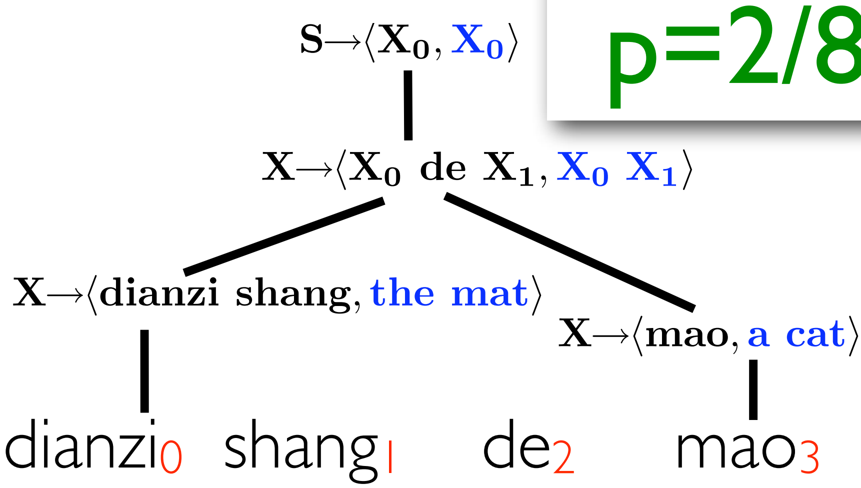
Compute Expectations over Products

expected translation length: 4.75

variance of translation length?

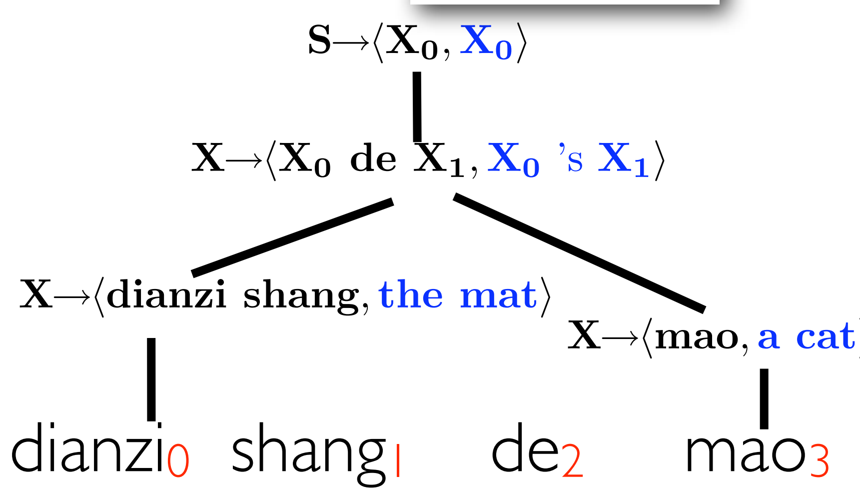
$2/8 \times (4-4.75)^2 + 6/8 \times (5-5.75)^2 \approx 0.56$

p=2/8



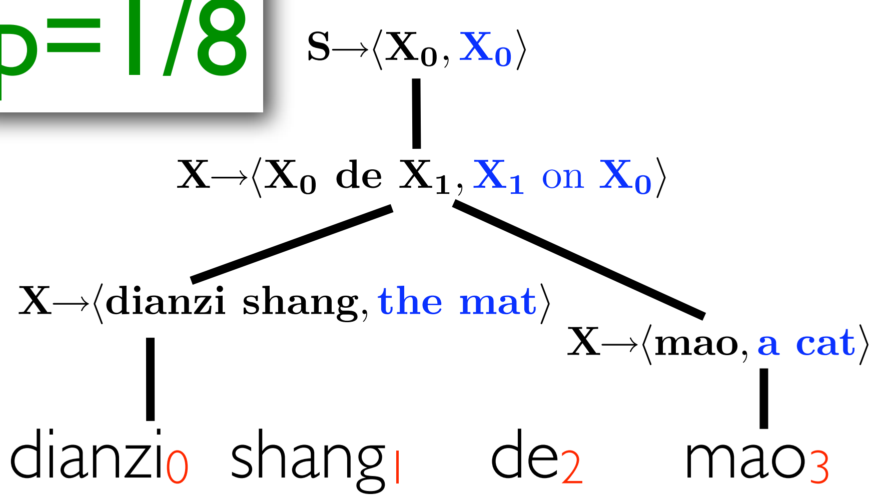
the mat a cat

p=3/8



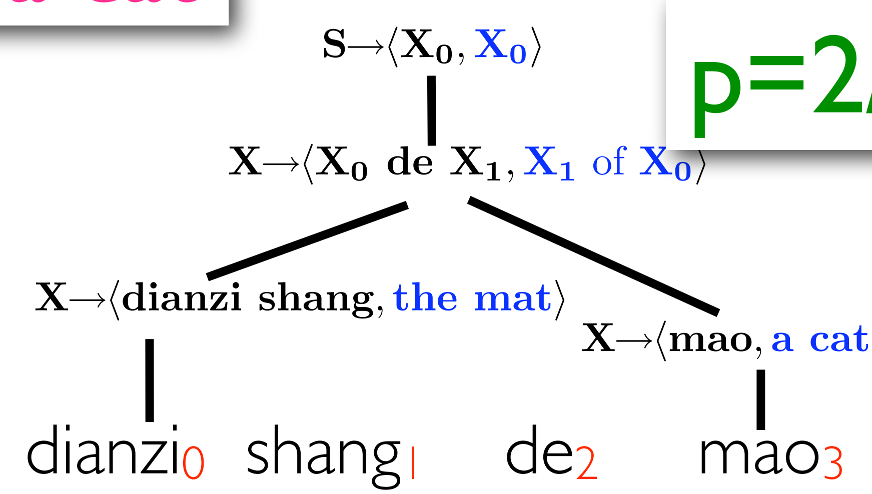
the mat 's a cat

p=1/8



a cat on the mat

p=2/8



a cat of the mat

Compute the Variance in Translation Length

- choose a semiring

  second-order expectation semiring

- specify a weight for each edge

$$k_e \stackrel{\text{def}}{=} \langle p_e, p_e r_e, p_e s_e, p_e r_e s_e \rangle$$

p_e: transition probability or log-linear score at edge e

$r_e = s_e$: number of English words generated at edge e

- run the inside algorithm

Second-order Expectations on Hypergraphs

- Expectation of **products** over a hypergraph

$$\bar{t} \stackrel{\mathrm{def}}{=} \mathbb{E}_p[r \cdot s] = \sum_{d \in \mathrm{HG}} p(d) r(d) s(d)$$

- r and s are additively decomposed

$$r(d) \stackrel{\mathrm{def}}{=} \sum_{e \in d} r_e$$

$$s(d) \stackrel{\mathrm{def}}{=} \sum_{e \in d} s_e$$

r and s can be identical or different functions.

Applications of expectation semirings: a summary

First-order:

Quantity	Weight for edge e	Value at root
Expectation	$\langle p_e, p_e r_e \rangle$	$\langle Z, \overline{r} \rangle$
First-order gradient	$\langle p_e, \nabla p_e \rangle$	$\langle Z, \nabla Z \rangle$

Second-order:

Covariance matrix	$\langle p_e, p_e r_e, p_e s_e, p_e r_e s_e \rangle$	$\langle Z, \overline{r}, \overline{s}, \overline{t} \rangle$
Hessian matrix	$\langle p_e, \nabla p_e, \nabla p_e, \nabla^2 p_e \rangle$	$\langle Z, \nabla Z, \nabla Z, \nabla^2 Z \rangle$
Gradient of expectation	$\langle p_e, p_e r_e, \nabla p_e, (\nabla p_e) r_e + p_e (\nabla r_e) \rangle$	$\langle Z, \overline{r}, \nabla Z, \nabla \overline{r} \rangle$

$$p_e = \exp(\Phi_e \cdot \theta) \qquad \nabla p_e = p_e \Phi_e$$

- choose a semiring
- define a weight for each edge
- run inside algorithm (or inside-outside for speedup)

Applications of expectation semirings: a summary

First-order:

Quantity	Weight for edge e	Value at root
Expectation	$\langle p_e, p_e r_e \rangle$	$\langle Z, \overline{r} \rangle$
First-order gradient	$\langle p_e, \nabla p_e \rangle$	$\langle Z, \nabla Z \rangle$

Second-order:

Covariance matrix	$\langle p_e, p_e r_e, p_e s_e, p_e r_e s_e \rangle$	$\langle Z, \overline{r}, \overline{s}, \overline{t} \rangle$
Hessian matrix	$\langle p_e, \nabla p_e, \nabla p_e, \nabla^2 p_e \rangle$	$\langle Z, \nabla Z, \nabla Z, \nabla^2 Z \rangle$
Gradient of expectation	$\langle p_e, p_e r_e, \nabla p_e, (\nabla p_e) r_e + p_e(\nabla r_e) \rangle$	$\langle Z, \overline{r}, \nabla Z, \nabla \overline{r} \rangle$

Entropy is an expectation!

- choose a semiring
- define a weight for each edge
- run inside algorithm (or inside-outside for speedup)

Applications of expectation semirings: a summary

First-order:

Quantity	Weight for edge e	Value at root
Expectation	$\langle p_e, p_e r_e \rangle$	$\langle Z, \overline{r} \rangle$
First-order gradient	$\langle p_e, \nabla p_e \rangle$	$\langle Z, \nabla Z \rangle$

Second-order:

Covariance matrix	$\langle p_e, p_e r_e, p_e s_e, p_e r_e s_e \rangle$	$\langle Z, \overline{r}, \overline{s}, \overline{t} \rangle$
Hessian matrix	$\langle p_e, \nabla p_e, \nabla p_e, \nabla^2 p_e \rangle$	$\langle Z, \nabla Z, \nabla Z, \nabla^2 Z \rangle$
Gradient of expectation	$\langle p_e, p_e r_e, \nabla p_e, (\nabla p_e) r_e + p_e (\nabla r_e) \rangle$	$\langle Z, \overline{r}, \nabla Z, \nabla \overline{r} \rangle$

Bayes risk is an expectation!

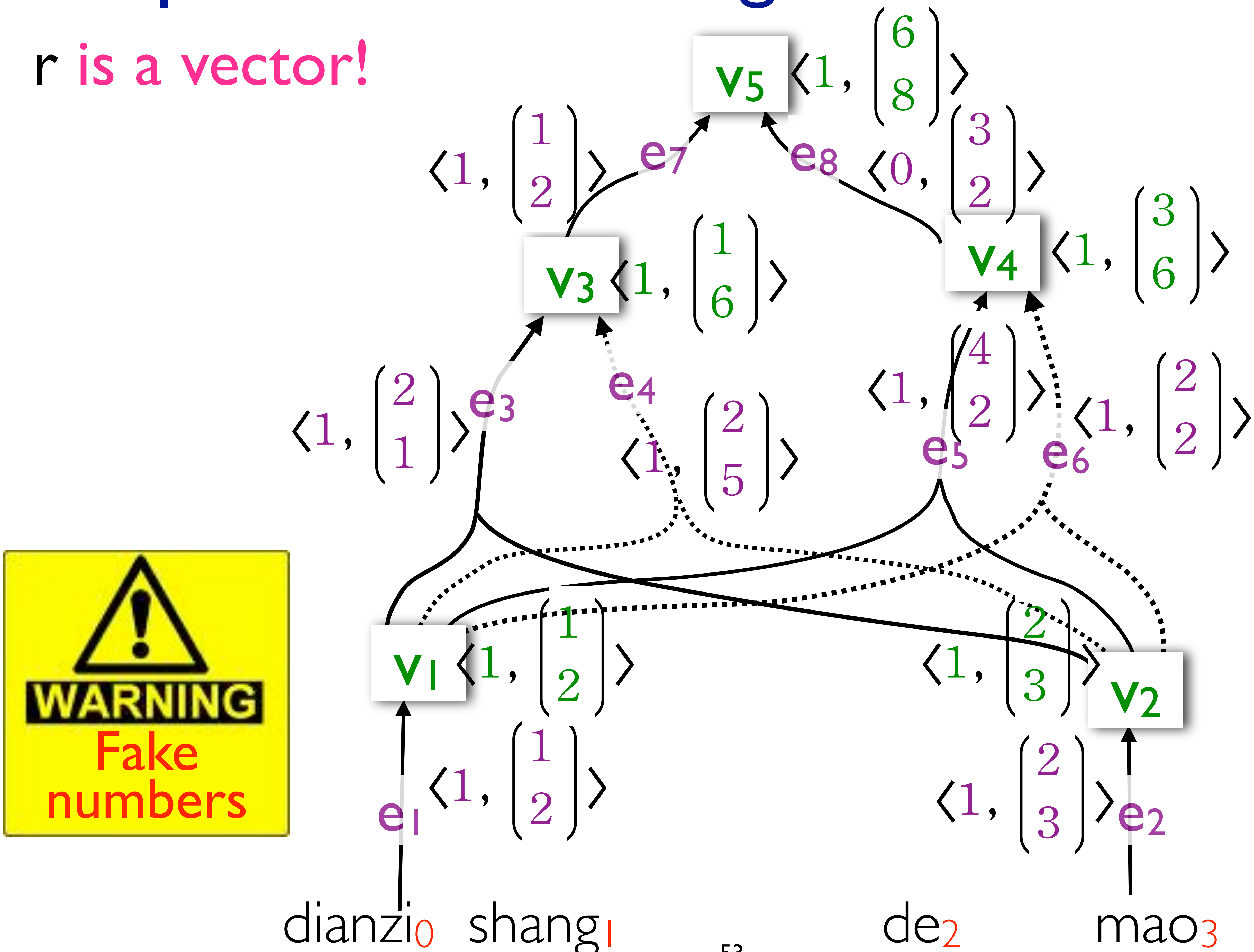
- choose a semiring
- define a weight for each edge
- run inside algorithm (or inside-outside for speedup)

Inside-Outside Speedup

Why is it slow?

Expectation Semirings with Vectors

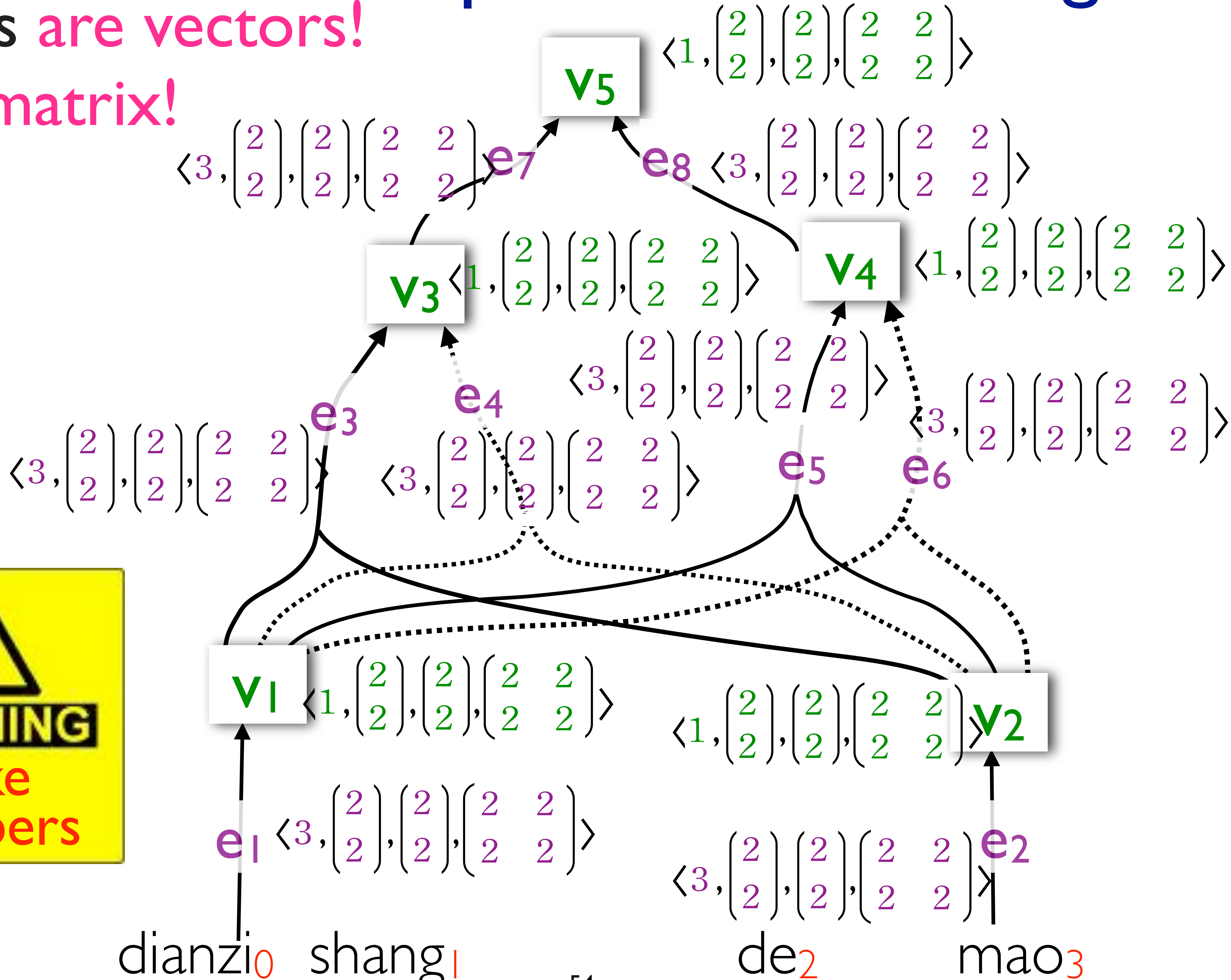
r is a vector!



Second-order Expectation Semirings

r and s are vectors!

t is a matrix!



Inside-Outside Speedup

See the paper for details!

Minimum Risk Training over Translation Forests

Minimum Risk Training

Finding optimal parameters:

$$\theta^* = \mathit{argmin}\ \text{Risk}(P_\theta) - \text{Temperature} \times \text{Entropy}(P_\theta)$$

- P_θ is a probability distribution over trees, parameterized by the parameters θ

Why Minimum Risk Training?

	Min-Risk	MERT	CRF	Perceptron	MIRA
Gradient descent	☺	☹	☺	?	?
BLEU	☺	☺	☹	☹	☺
Latent variable	☺	☹	☺	☹	☹
Oracle translation	☺	☺	☹	☹	☹
Model regularization	☺	☹	☺	☺	☺

Experimental Results

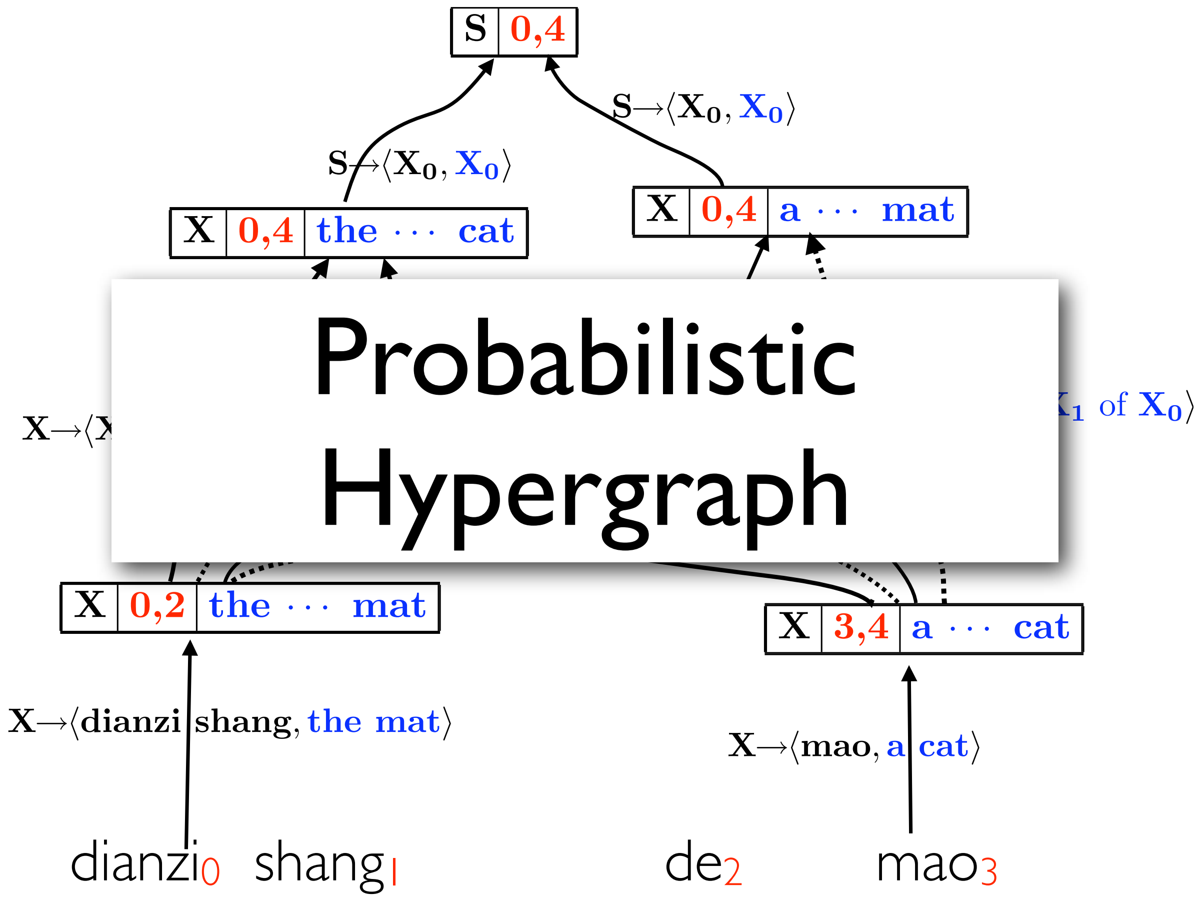
- Data set: IWSLT CN-EN 2005

	Training scheme	test
(Och, 2002)	MERT (Nbest, small)	47.7
(Smith and Eisner, 2006)	MR (Nbest, small)	47.7
NEW!	MR (hypergraph, small)	48.4
NEW!	MR (hypergraph, large)	**48.7**

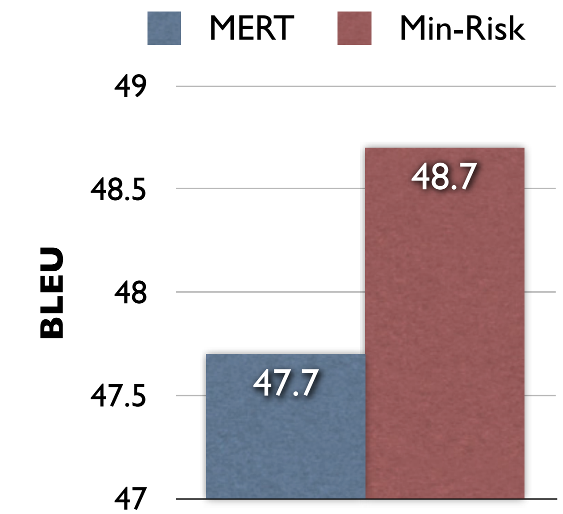
Small: five features as in regular MERT

Large: two-stage forest-reranking with **21k** additional target-side ngram features

Summary



Improved BLEU score!



This work provides a unified, elegant, and efficient framework in computing all of these!

- First-order quantities:
  - expectation
    - entropy
    - cross-entropy
    - KL divergence
    - Bayes risk
    - feature expectations
  - first-order gradient of Z

- Second-order quantities:
  - Covariance matrix
    - feature interaction
  - Hessian matrix of Z
    - second-order gradient descent
  - gradient of expectation
    - gradient of entropy or Bayes risk

Future: machine learning for MT

feature interaction

second-order gradient descent

....

minimum risk

deterministic annealing

active learning

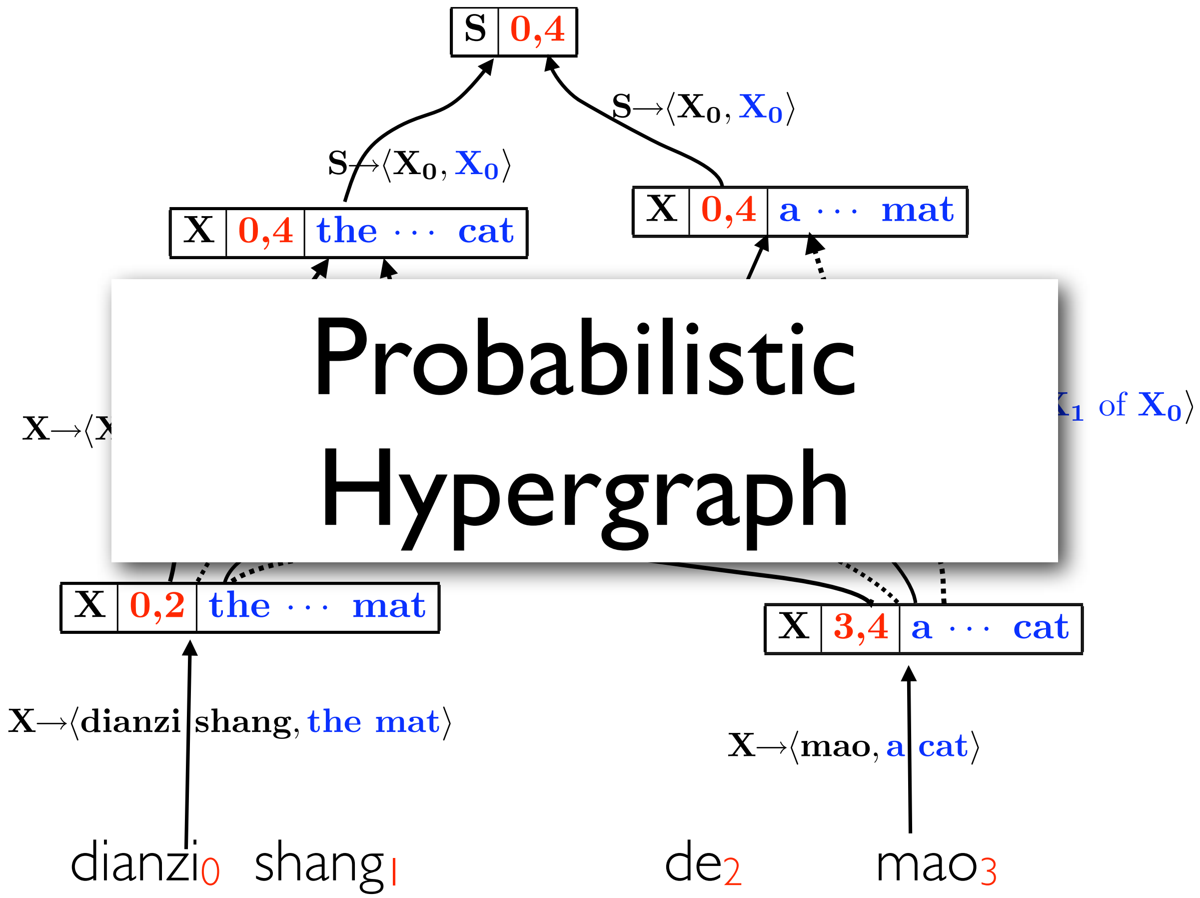
semi-supervised learning

semirings for parameter estimation

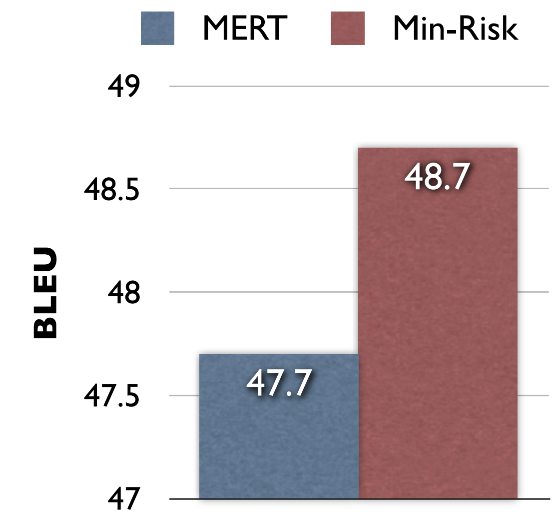

Joshua
semiring

Thank you!

谢谢！



Improved BLEU score!



This work provides a unified, elegant, and efficient framework in computing all of these!

- First-order quantities:
  - expectation
    - entropy
    - cross-entropy
    - KL divergence
    - Bayes risk
    - feature expectations
  - first-order gradient of Z

- Second-order quantities:
  - Covariance matrix
    - feature interaction
  - Hessian matrix of Z
    - second-order gradient descent
  - gradient of expectation
    - gradient of entropy or Bayes risk