Binarized Forest-to-Tree Translation

Authors

Hao Zhang	Google Research
Licheng Fang	University of Rochester
Peng Xu	Google Research
Xiaoyun Wu	Google Research

Presenter

Waleed Ammar

synchronous tree substitution grammar

Phrase based rule: {“green houses”,
“maisons vertes”}

STSG rule1: {np(adj(green) n(house)),
np(n(maisons) adj(vertes))}

STSG rule2: {np(adj$_0$ n$_1$), np(n$_1$ adj$_0$)}

STSG rule3: {np(adj$_0$ n$_1$), “{1} {0}”}

optimal forest-to-string translation

$$e = \mathcal{Y}\left(\mathop{\arg\max}_{d \in D(T),\, T \in F(f)} P(d|T) \right)$$

f := source sentence

F(f) := forest of parse trees of a source sentence

D(T) := synchronous derivations of a parse tree

e := best translation

Y := yield of a derivation

$\boxed{\text{parse-forest construction}}$

rule extraction

rule binarization

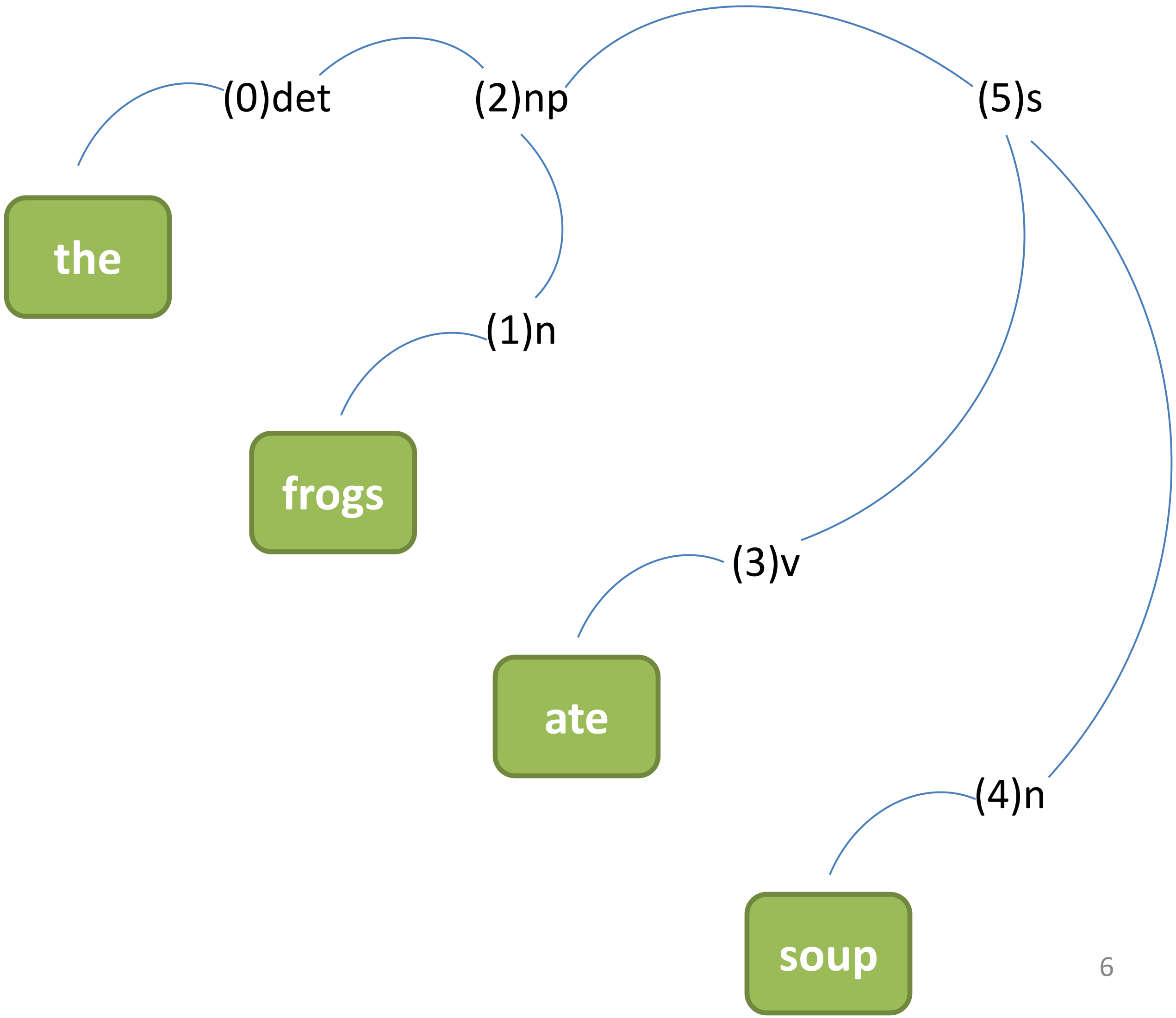
parse-forest construction

- Traditional forests
  - Construction
  - Drawbacks

- Proposed forests
  - Construction
  - Merits

parse-forest construction

best parse



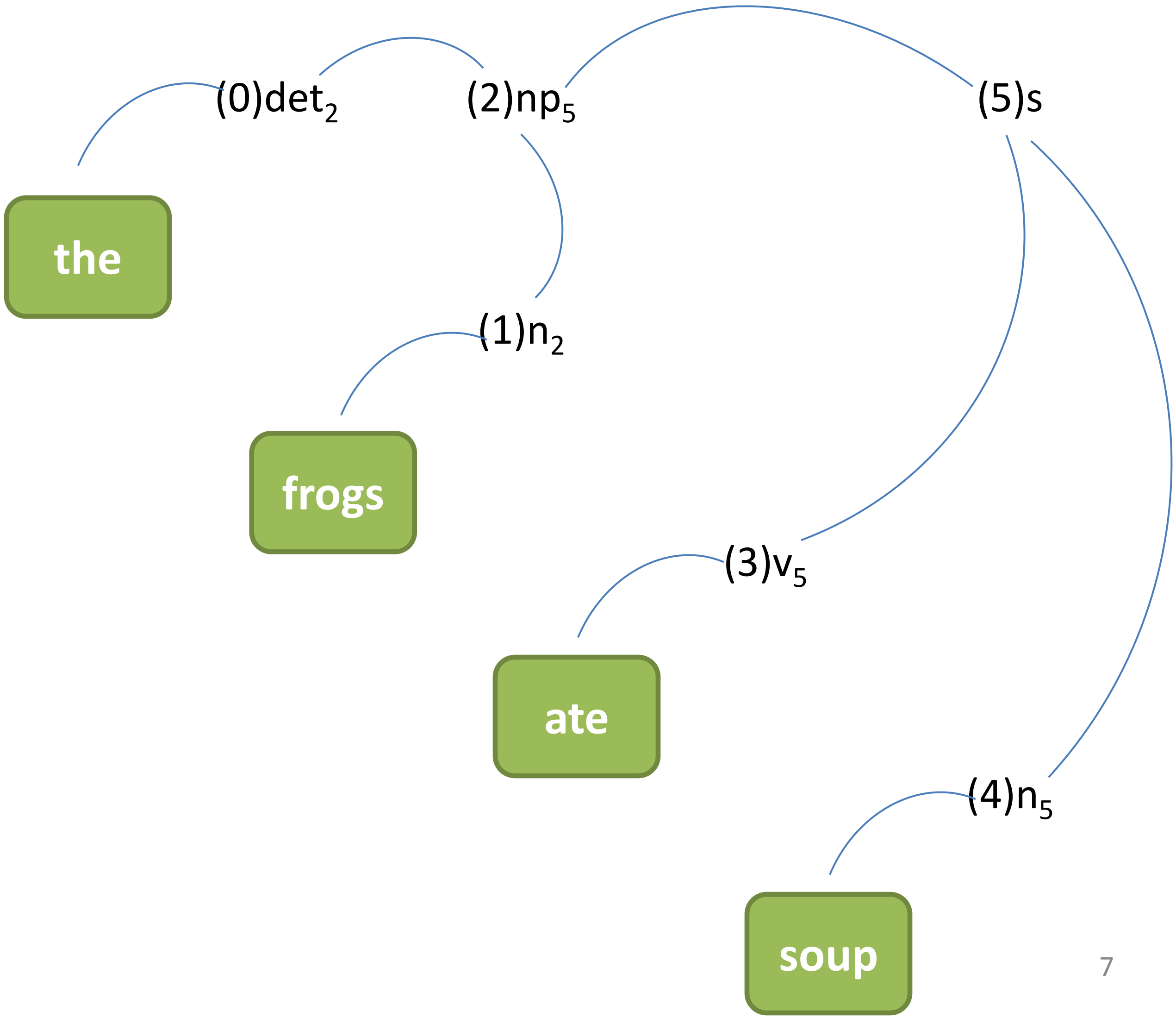
parse-forest construction

CYK-1

binarization:

ancestor

annotation



parse-forest construction

CYK-1

binarization:

drop

edges

the	(0)det_2	(2)np_5		(5)s
frogs		(1)n_2		
ate			(3)v_5	
soup				(4)n_5

parse-forest construction

CYK-1

binarization:

processing

order

the	(0)det_2	(2)np_5		(5)s
frogs		(1)n_2		
ate			(3)v_5	
soup				(4)n_5

parse-forest construction

CYK-1

binarization:

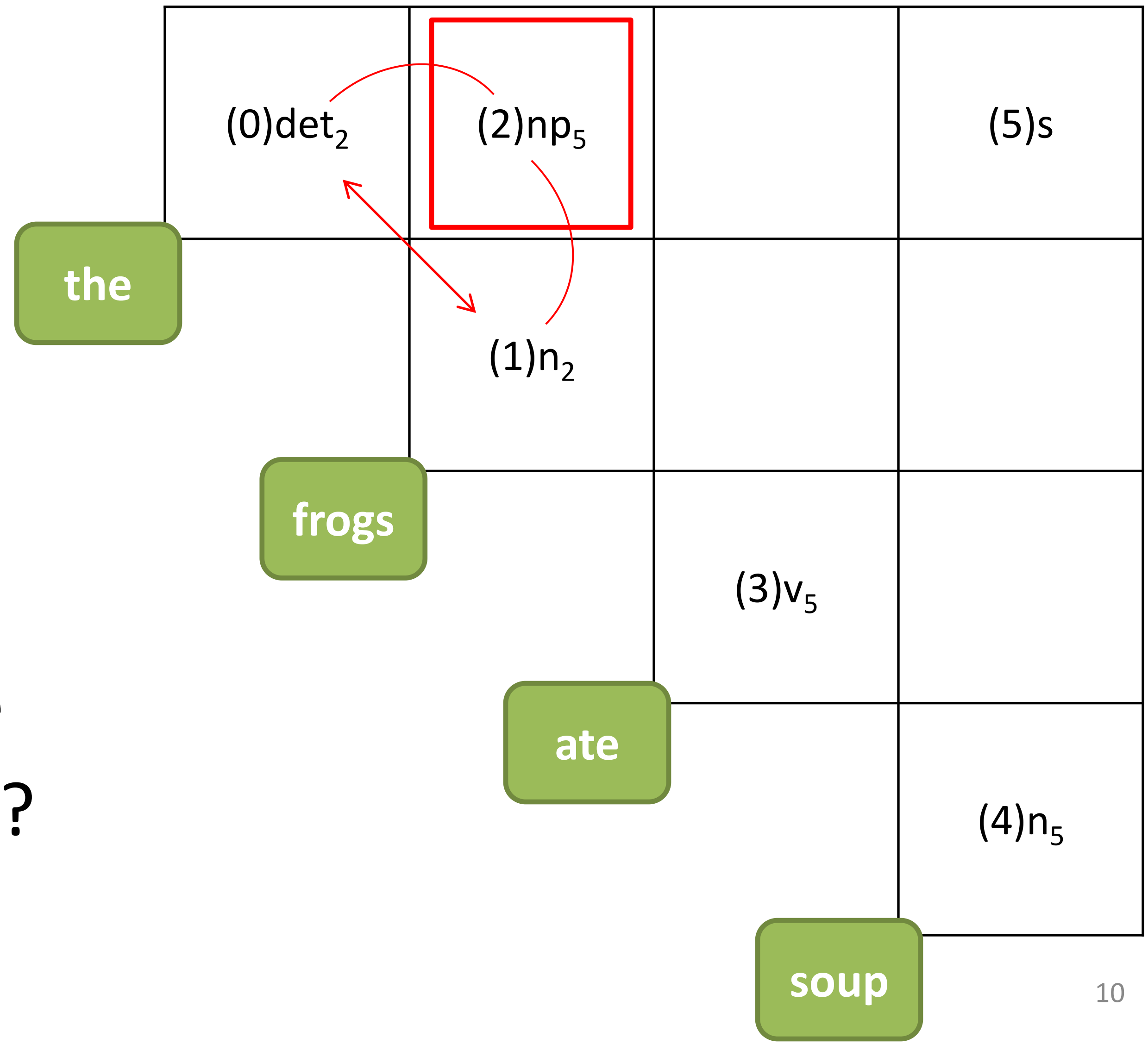
processing

at each node:

left & right have

shared ancestor?

Yes!



parse-forest construction

CYK-1

binarization:

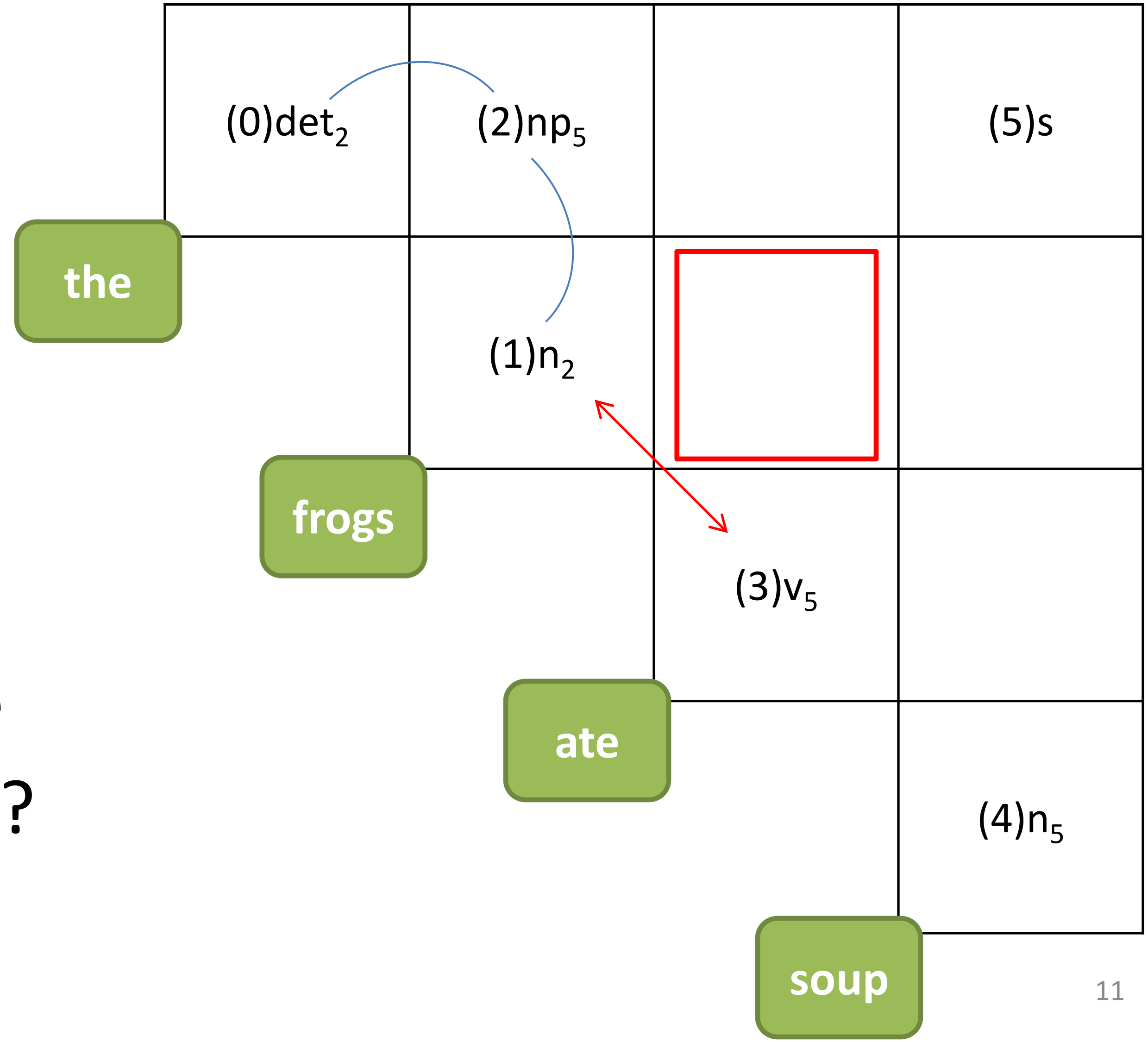
processing

at each node:

left & right have

shared ancestor?

No!



parse-forest construction

CYK-1

binarization:

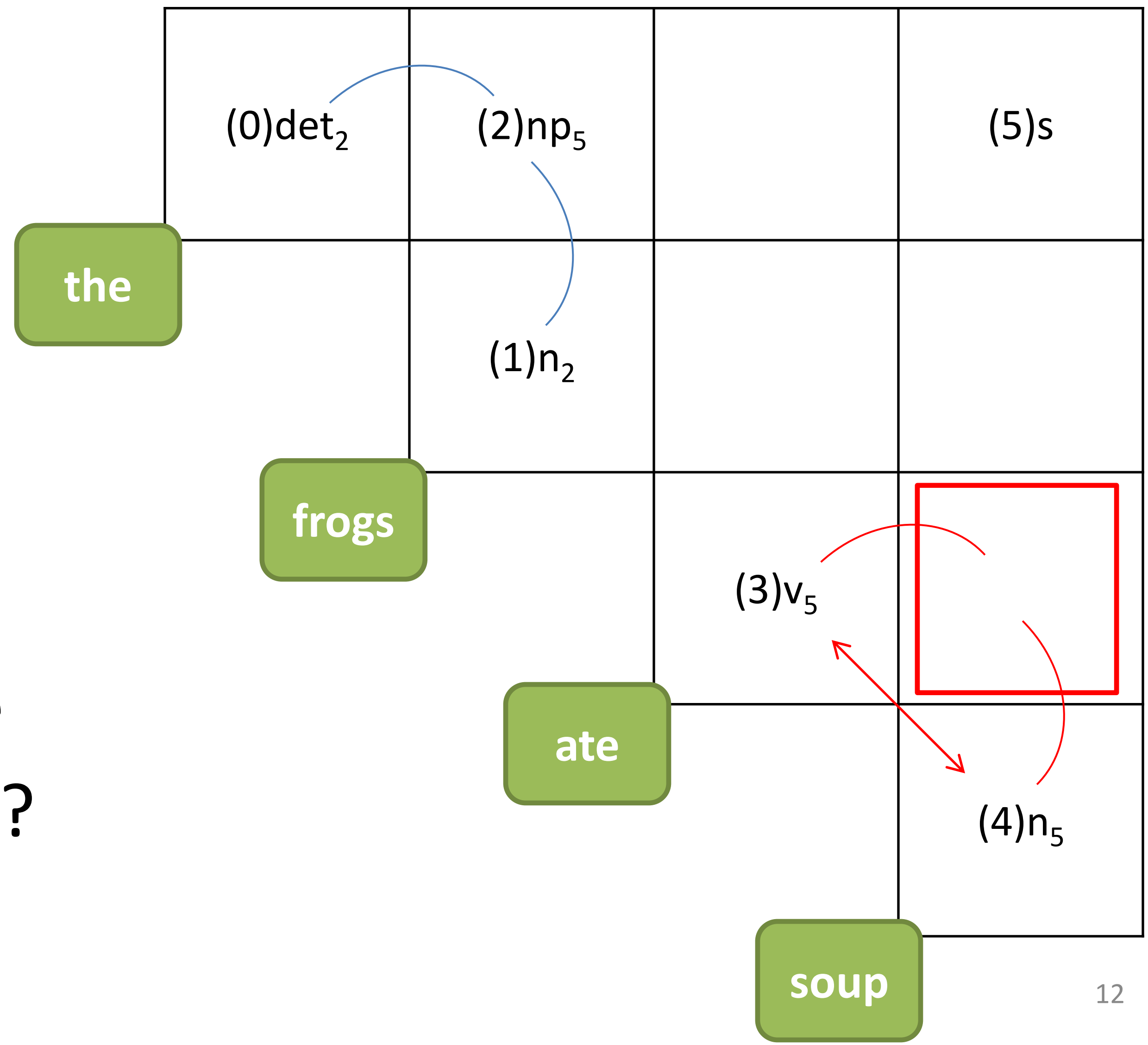
processing

at each node:

left & right have

shared ancestor?

Yes!

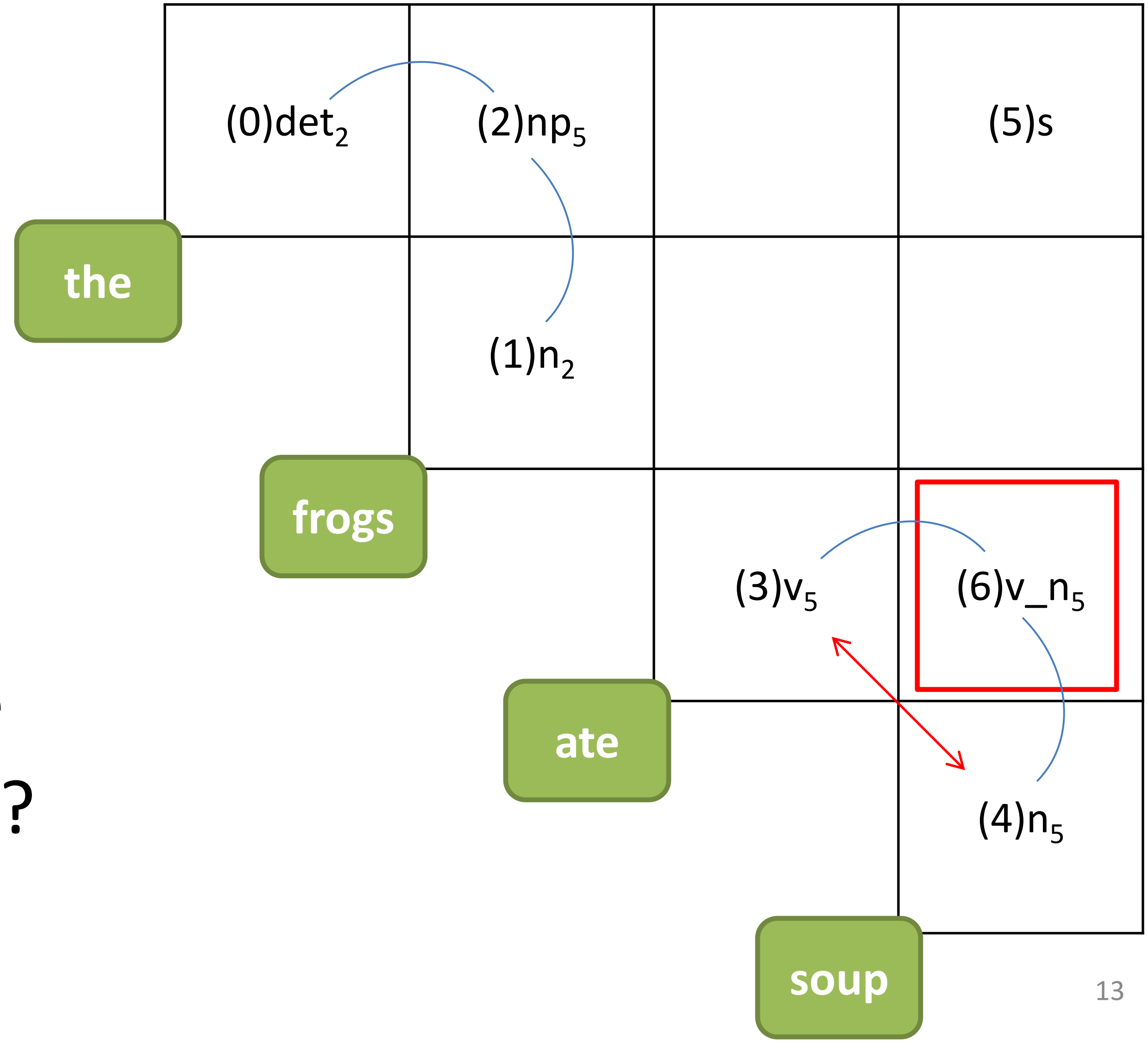


parse-forest construction

CYK-1 binarization:

processing at each node:

left & right have shared ancestor?
Yes!



parse-forest construction

CYK-1

binarization:

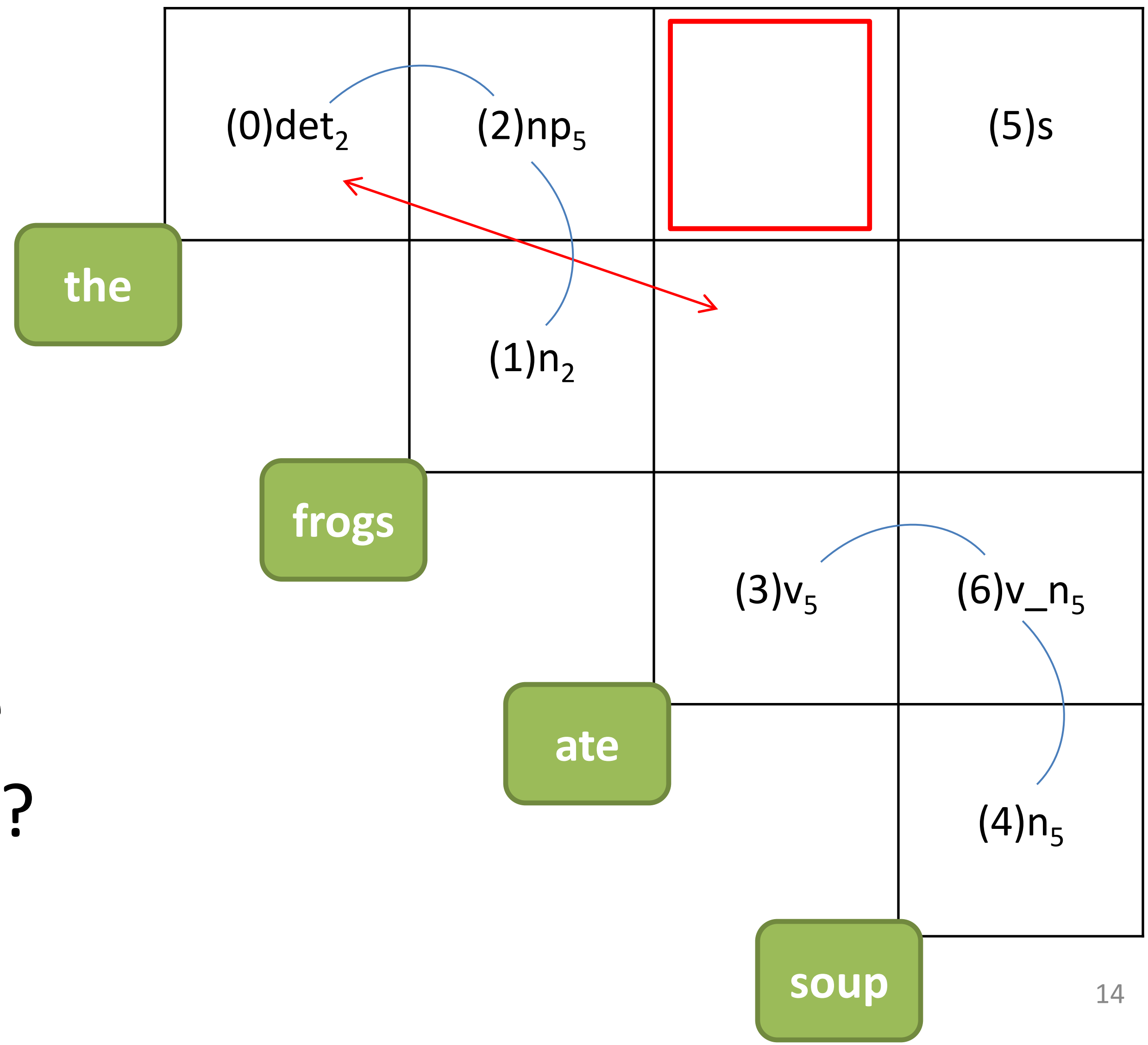
processing

at each node:

left & right have

shared ancestor?

No!



parse-forest construction

CYK-1

binarization:

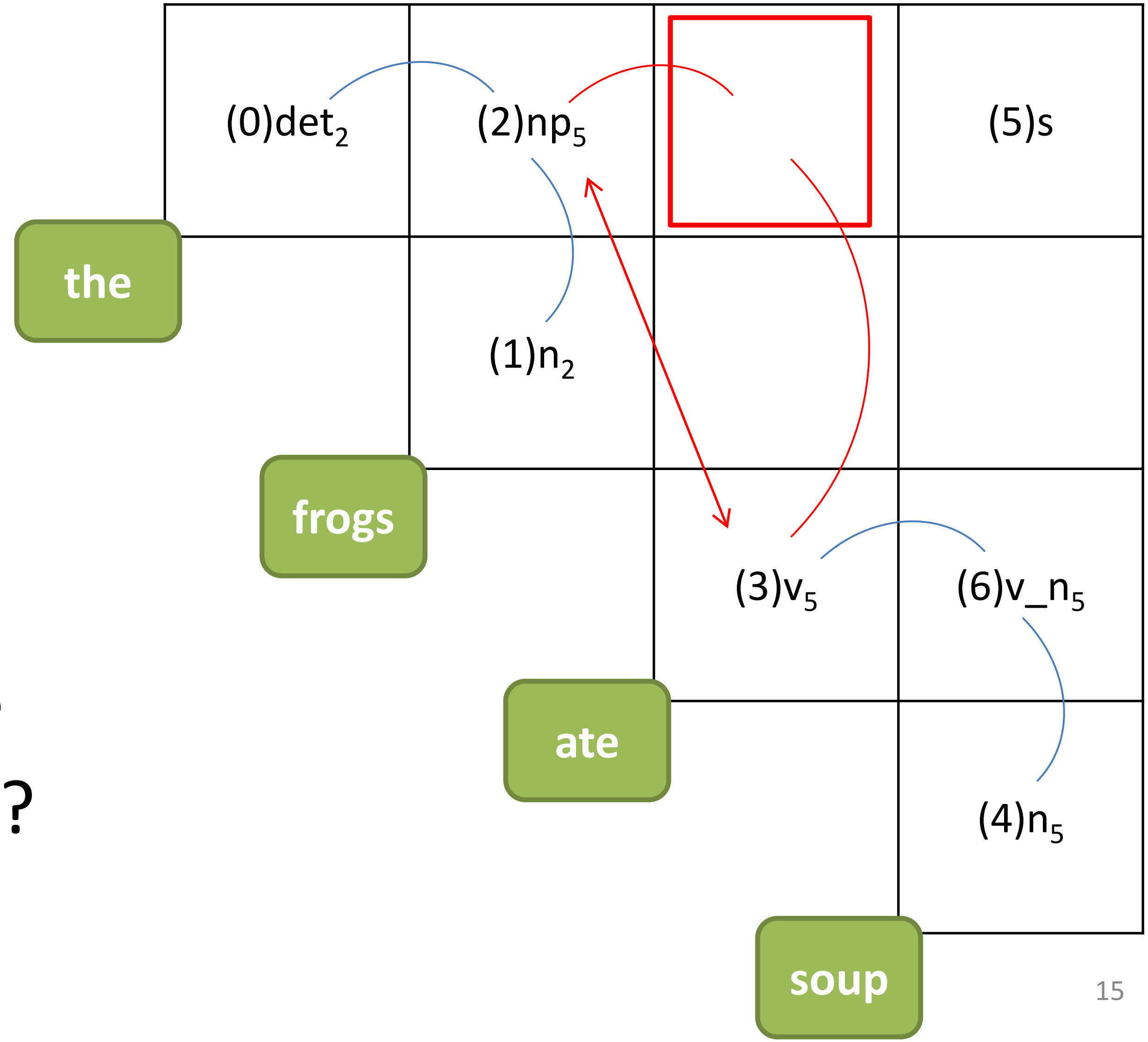
processing

at each node:

left & right have

shared ancestor?

Yes!



parse-forest construction

CYK-1

binarization:

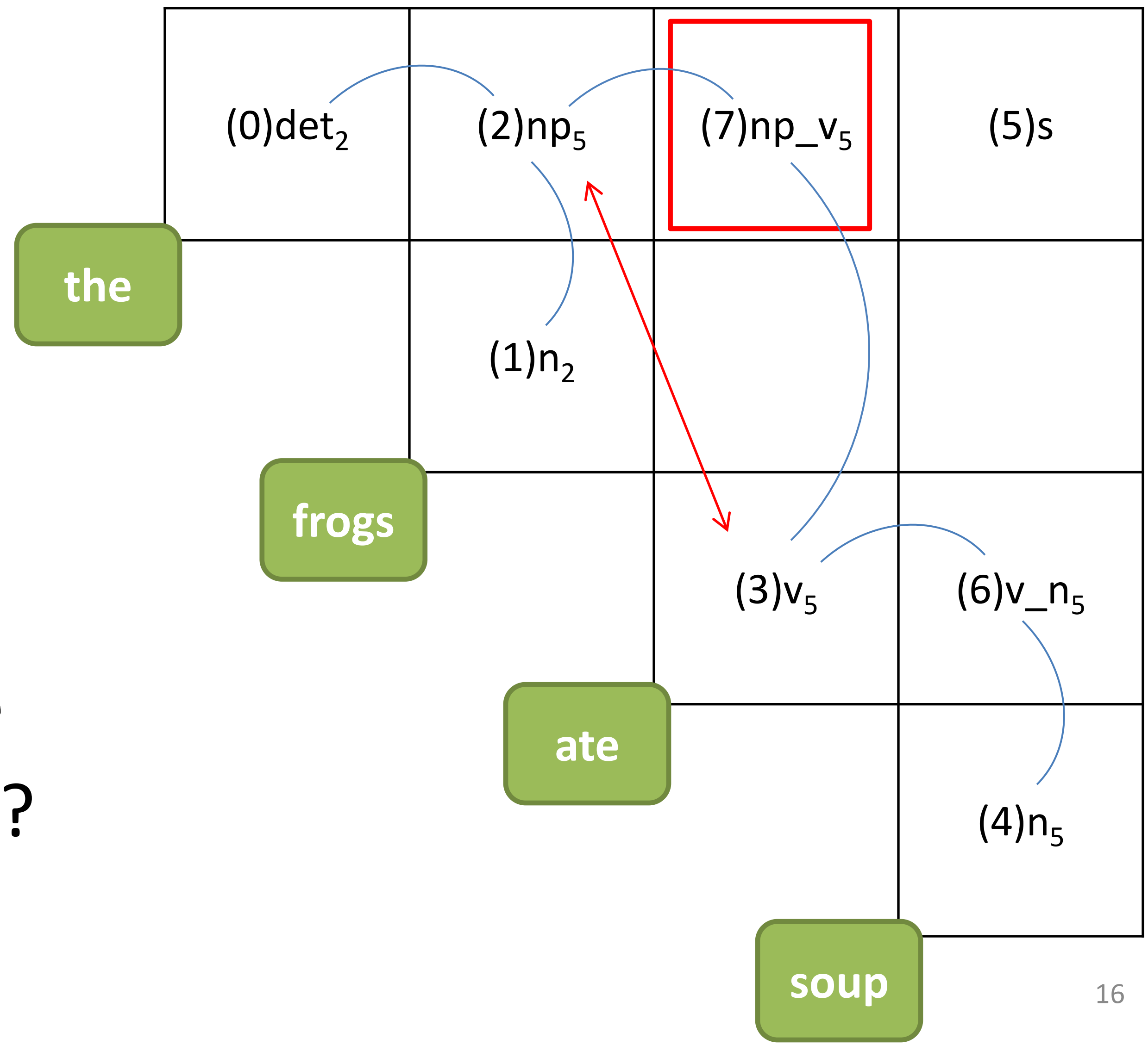
processing

at each node:

left & right have

shared ancestor?

Yes!



parse-forest construction

CYK-1

binarization:

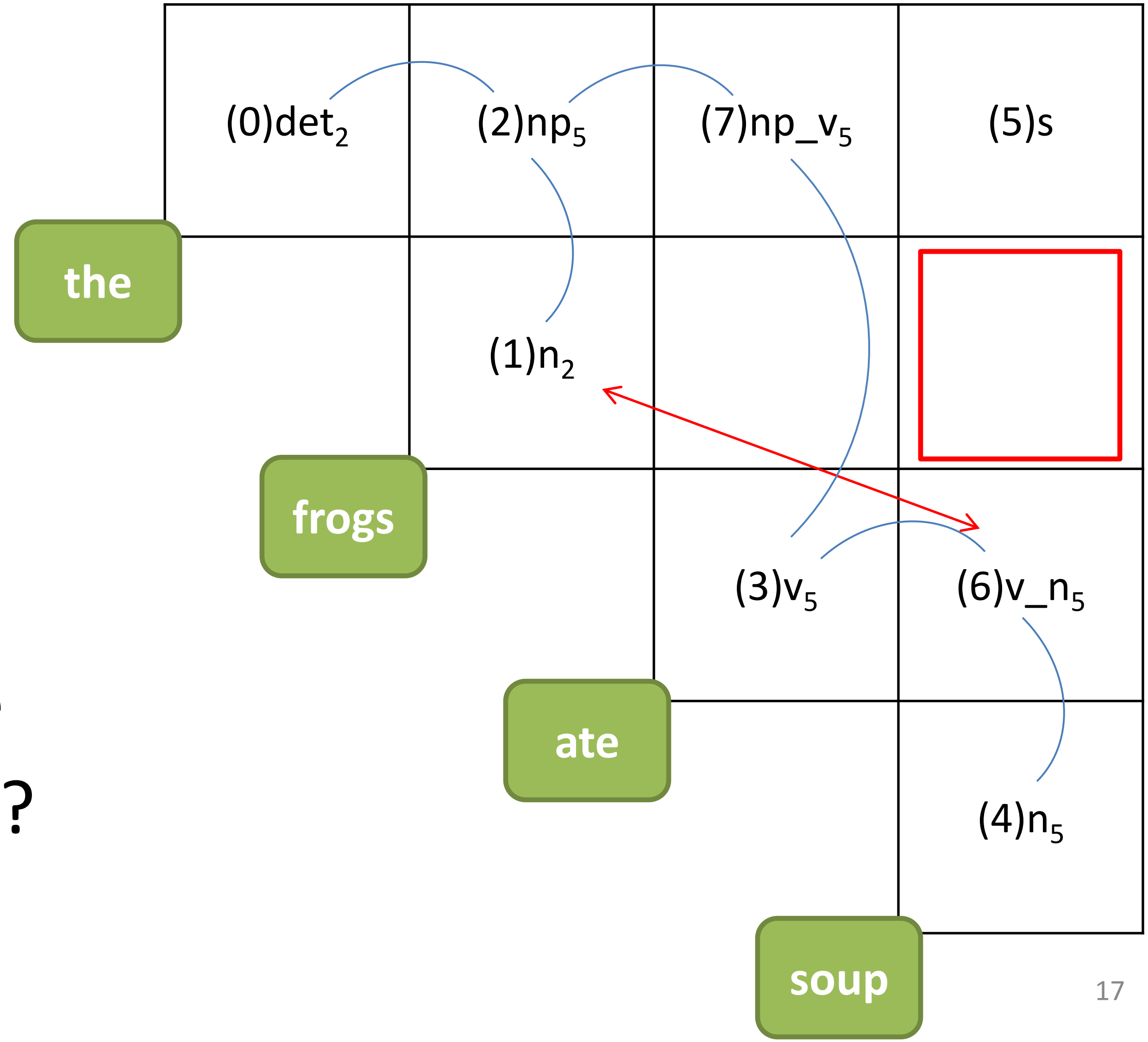
processing

at each node:

left & right have

shared ancestor?

No!

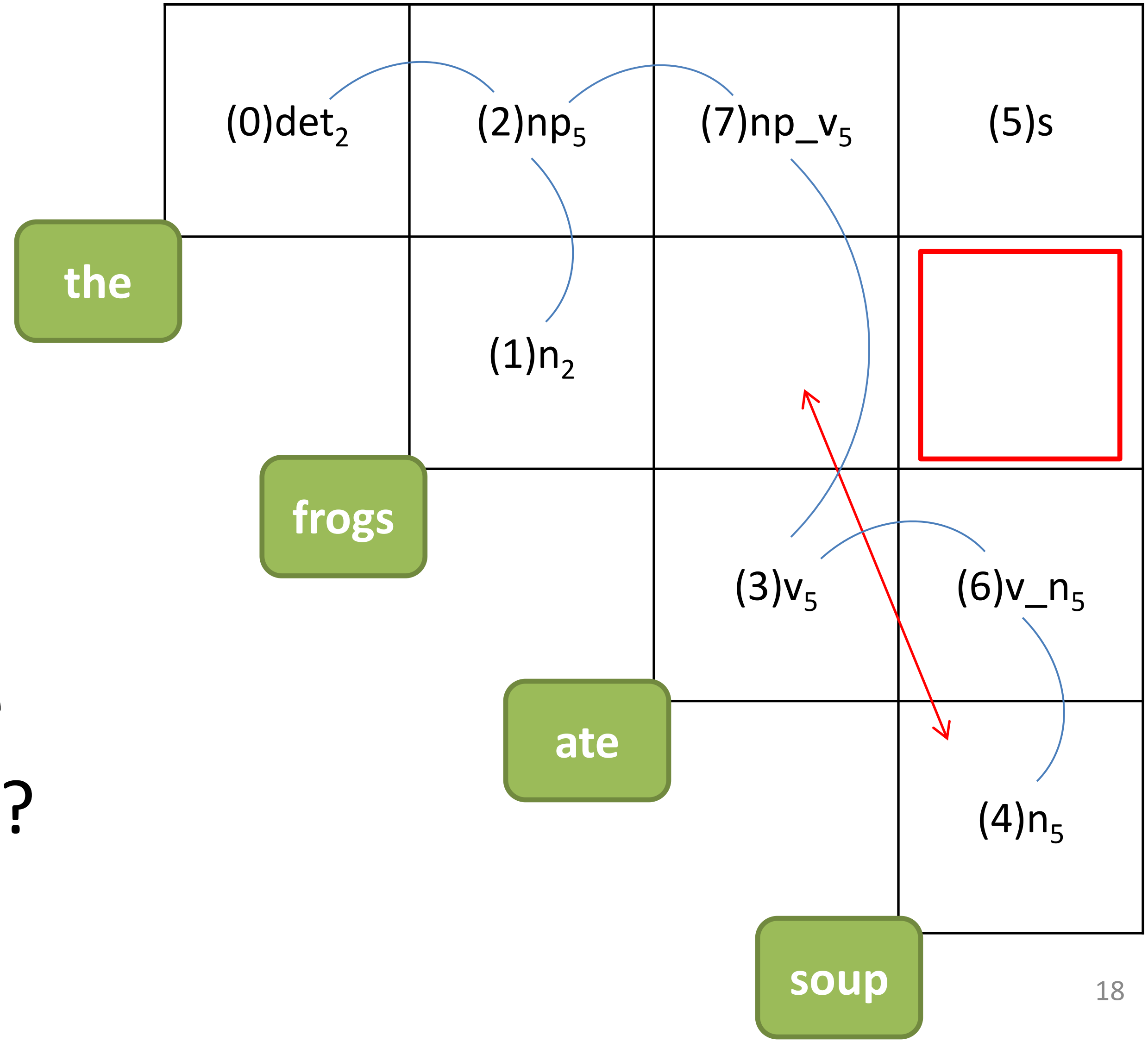


parse-forest construction

CYK-1 binarization:

processing at each node:

left & right have shared ancestor? No!



parse-forest construction

CYK-1

binarization:

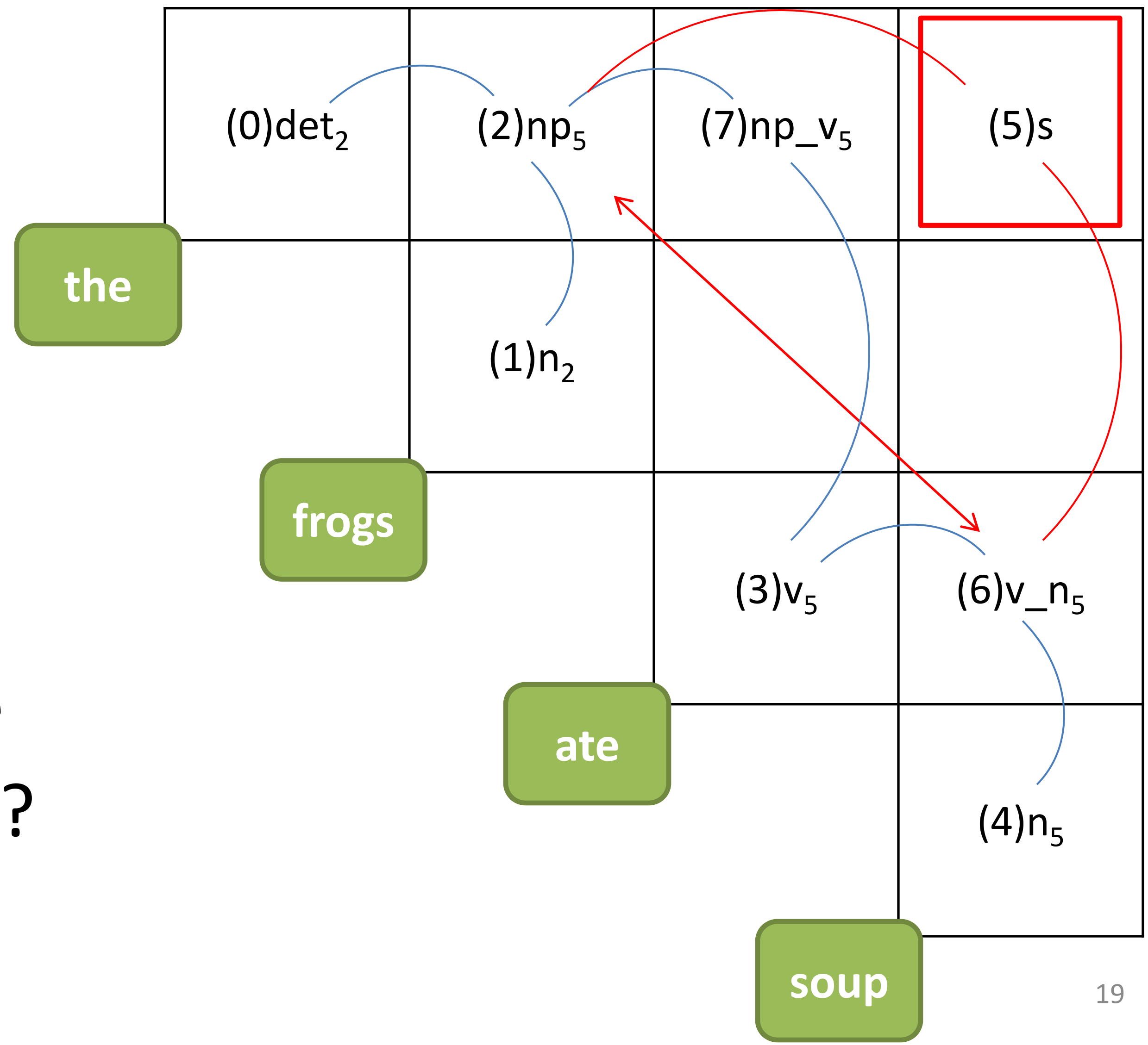
processing

at each node:

left & right have

shared ancestor?

Yes!



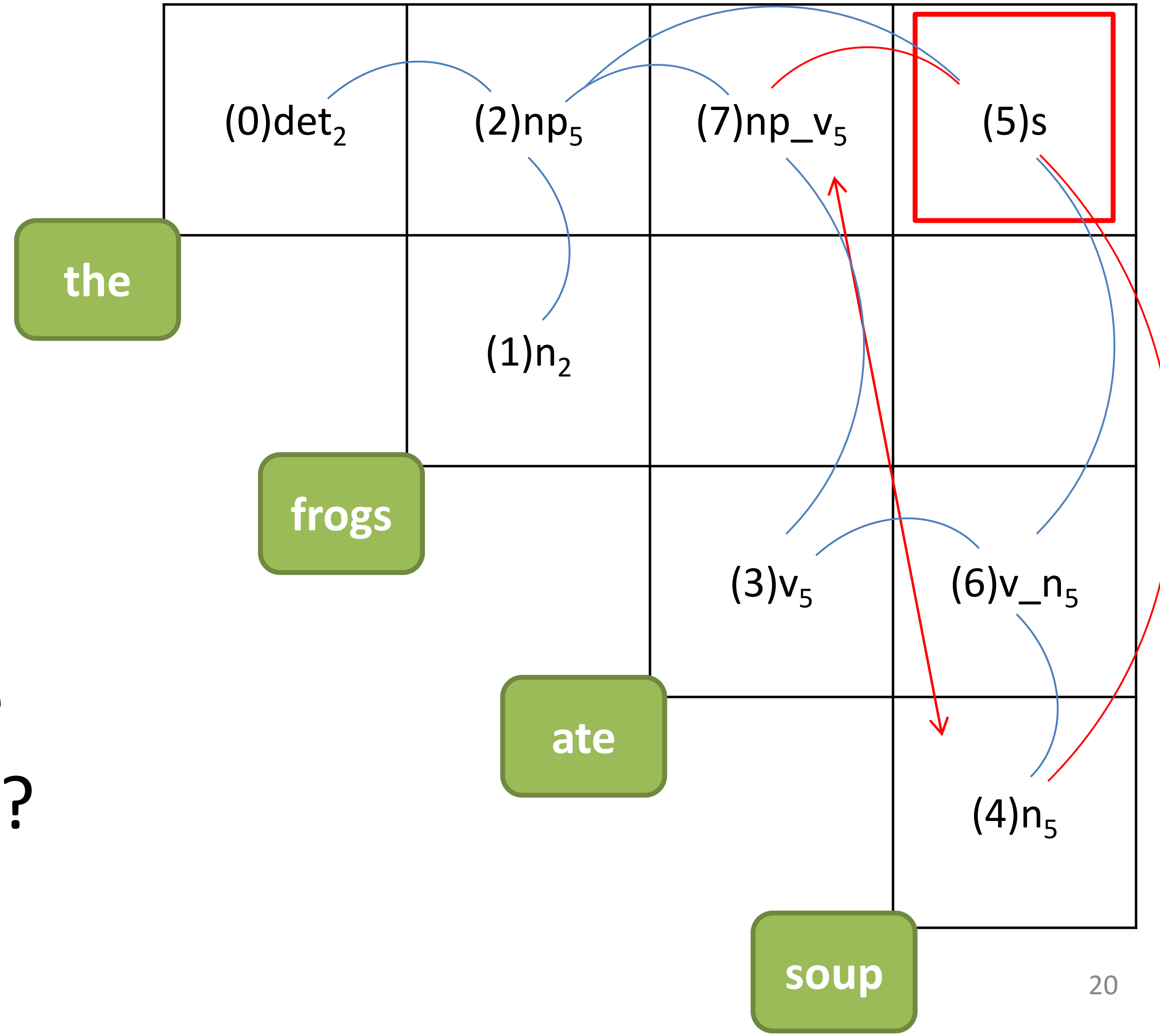
parse-forest construction

CYK-1

binarization:

processing
at each node:

left & right have
shared ancestor?
Yes!



parse-forest construction

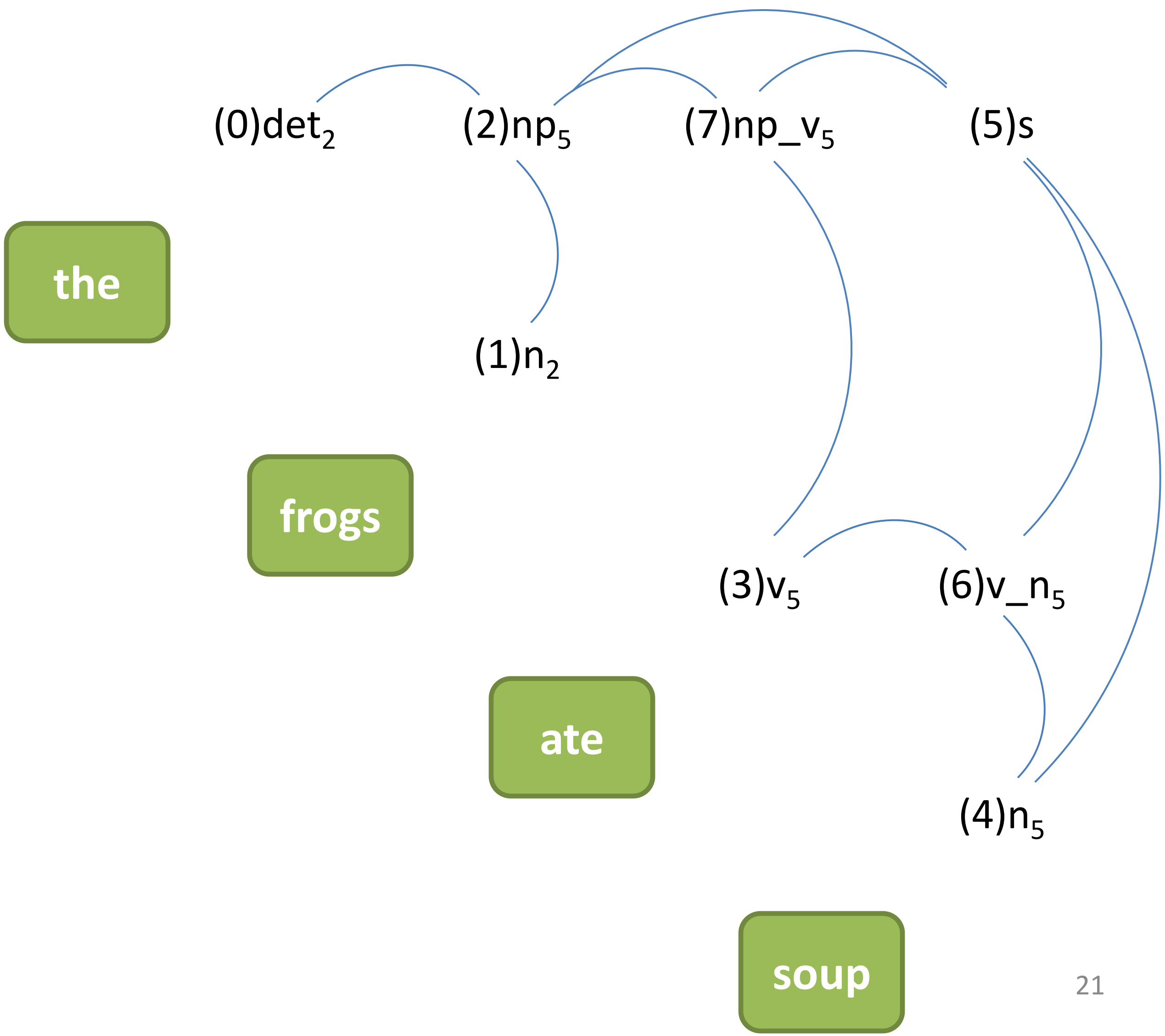
CYK-1

binarization:

final

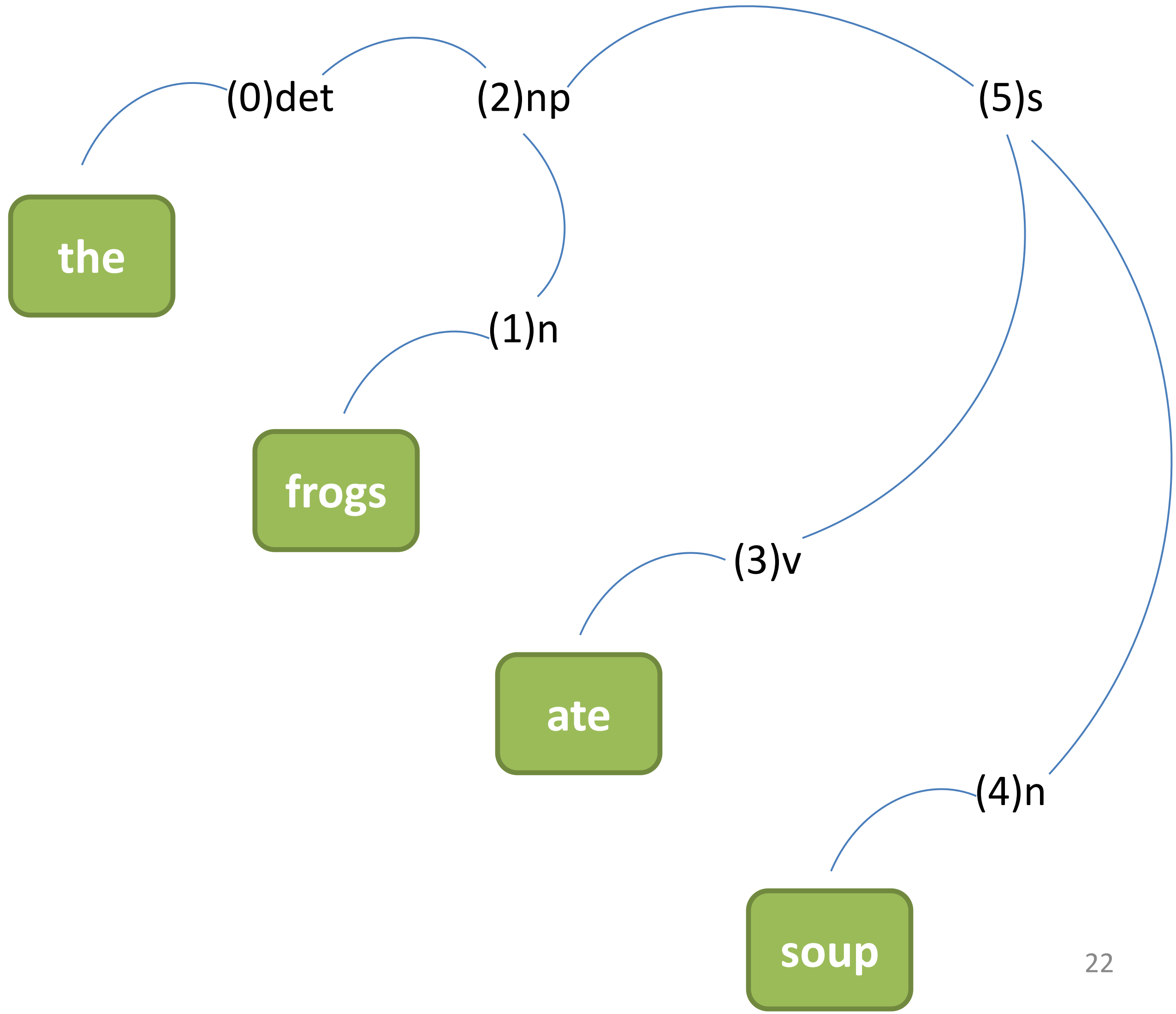
parsing

forest



parse-forest construction

best parse

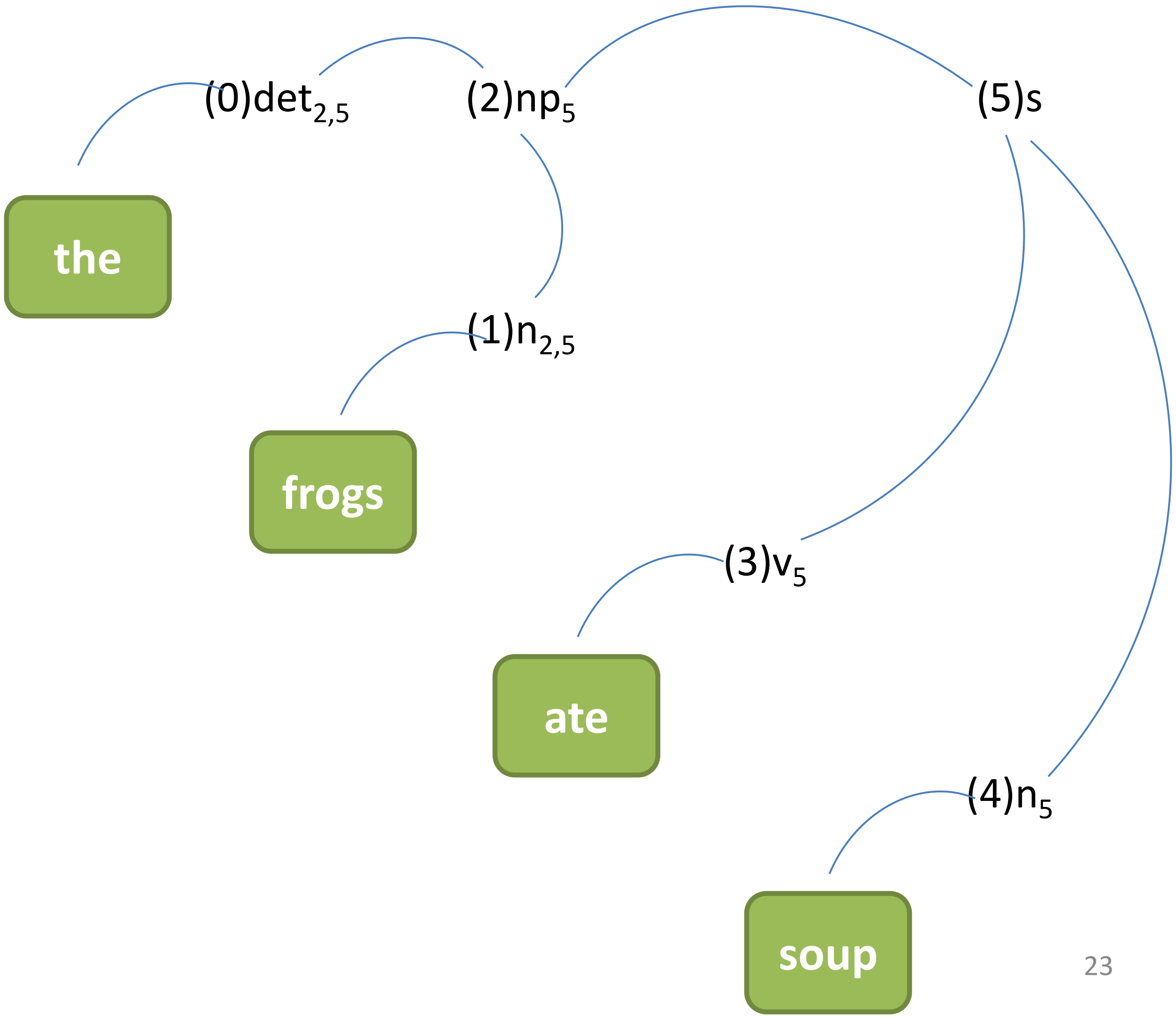


parse-forest construction

CYK-2

binarization:

ancestor annotation



parse-forest construction

CYK-2

binarization:

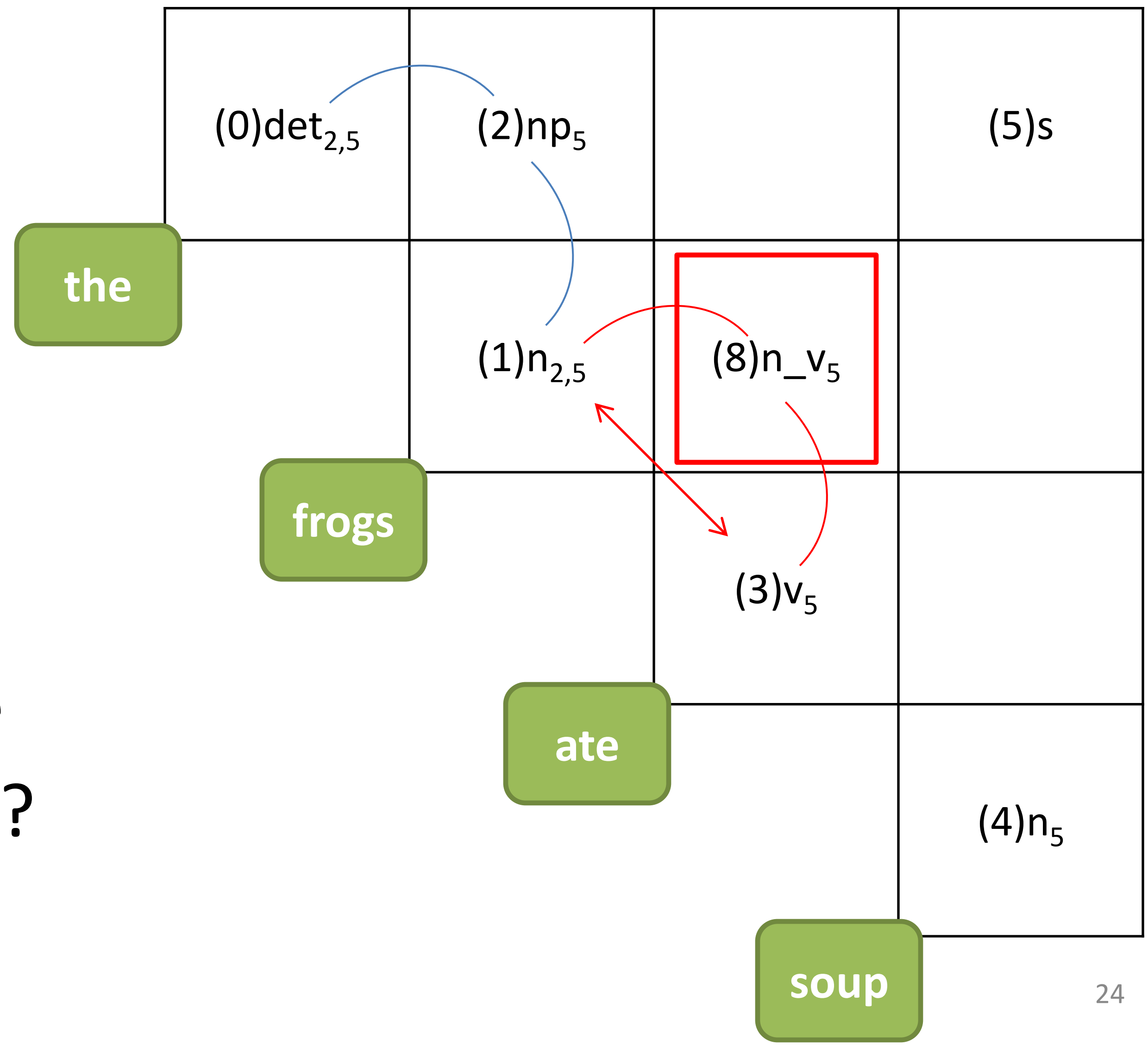
processing

at each node:

left & right have

shared ancestor?

Yes!



parse-forest construction

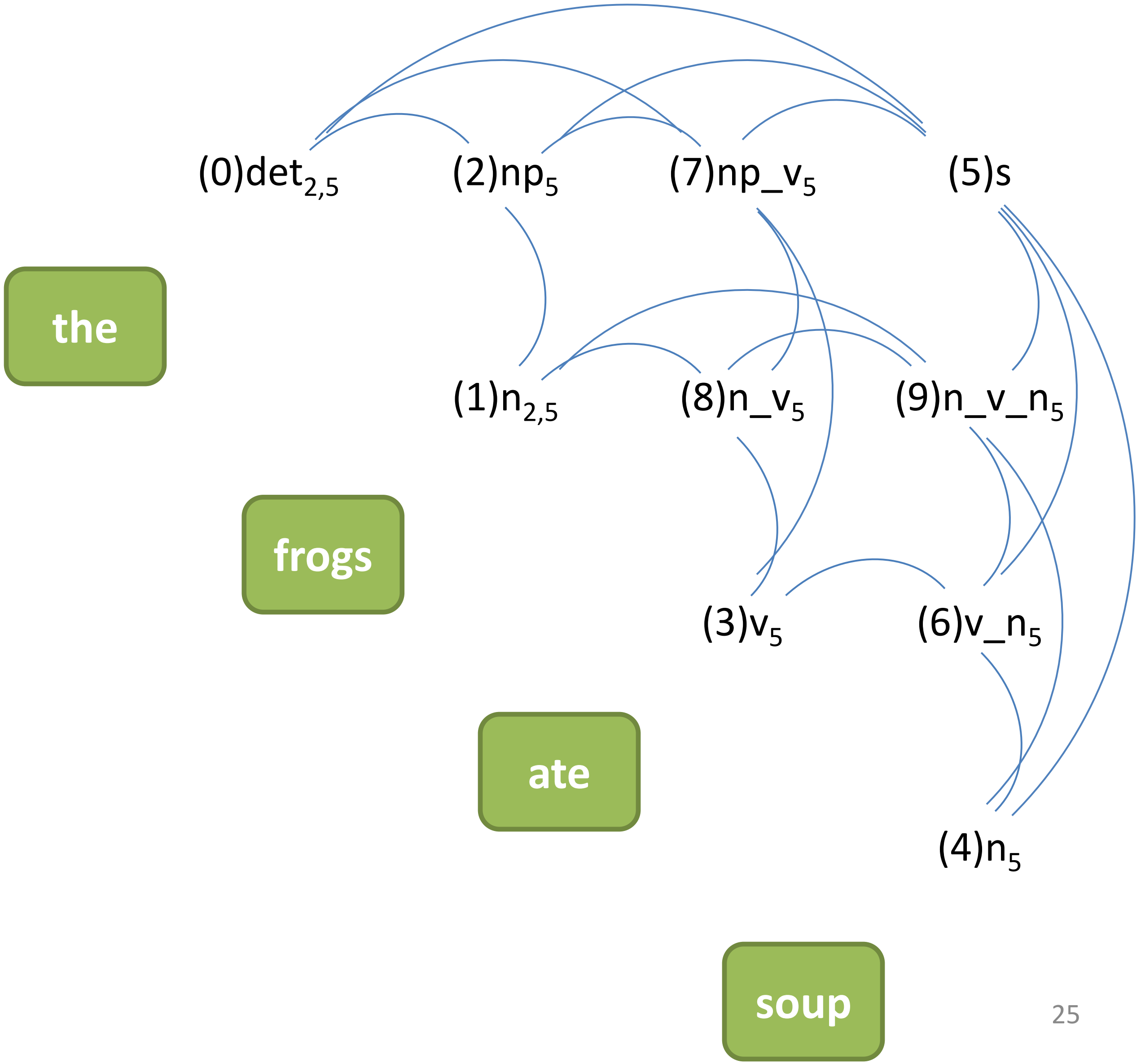
CYK-2

binarization:

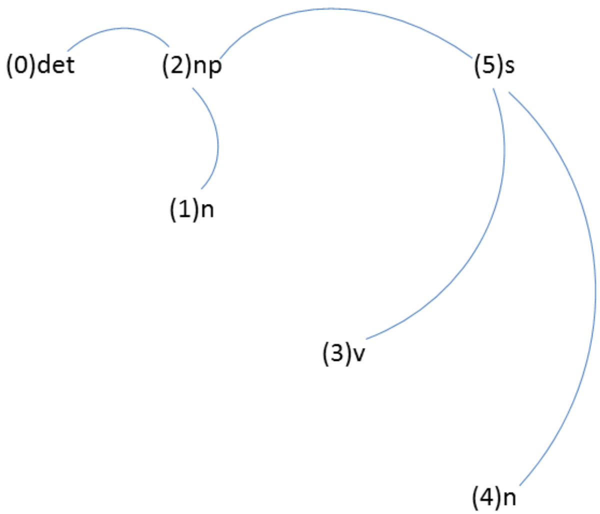
final

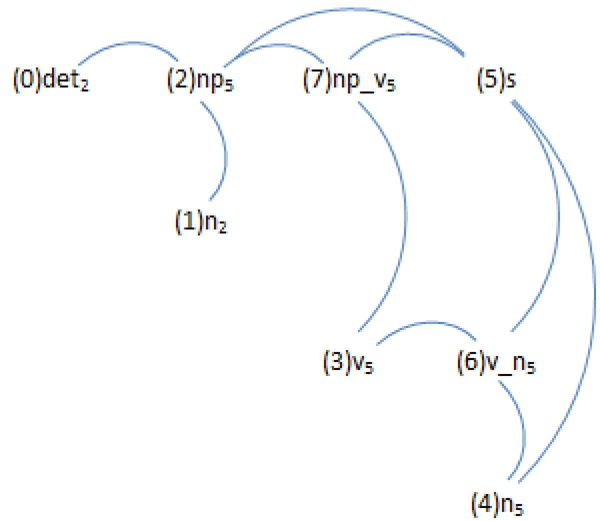
parsing

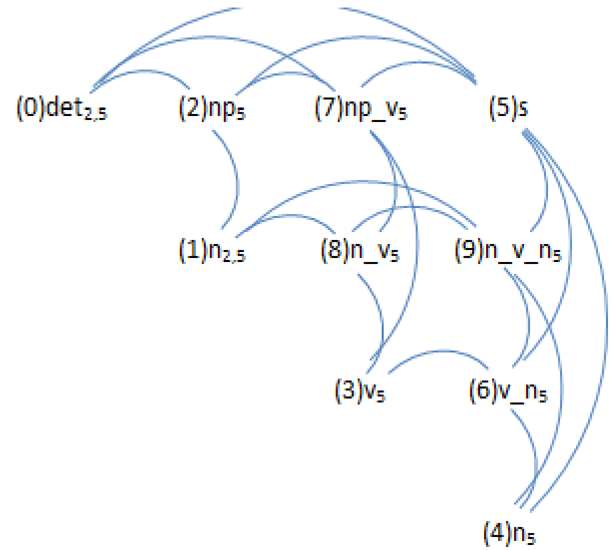
forest



parse-forest construction







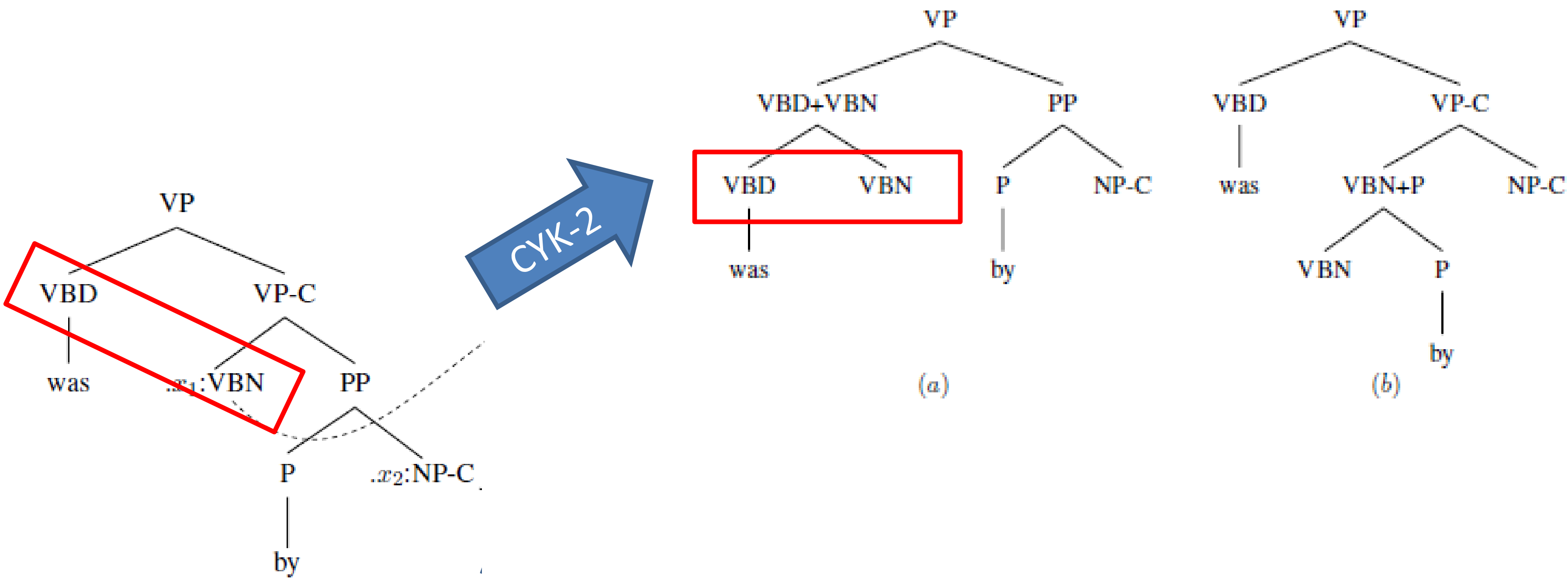
best parse

CYK-1 forest

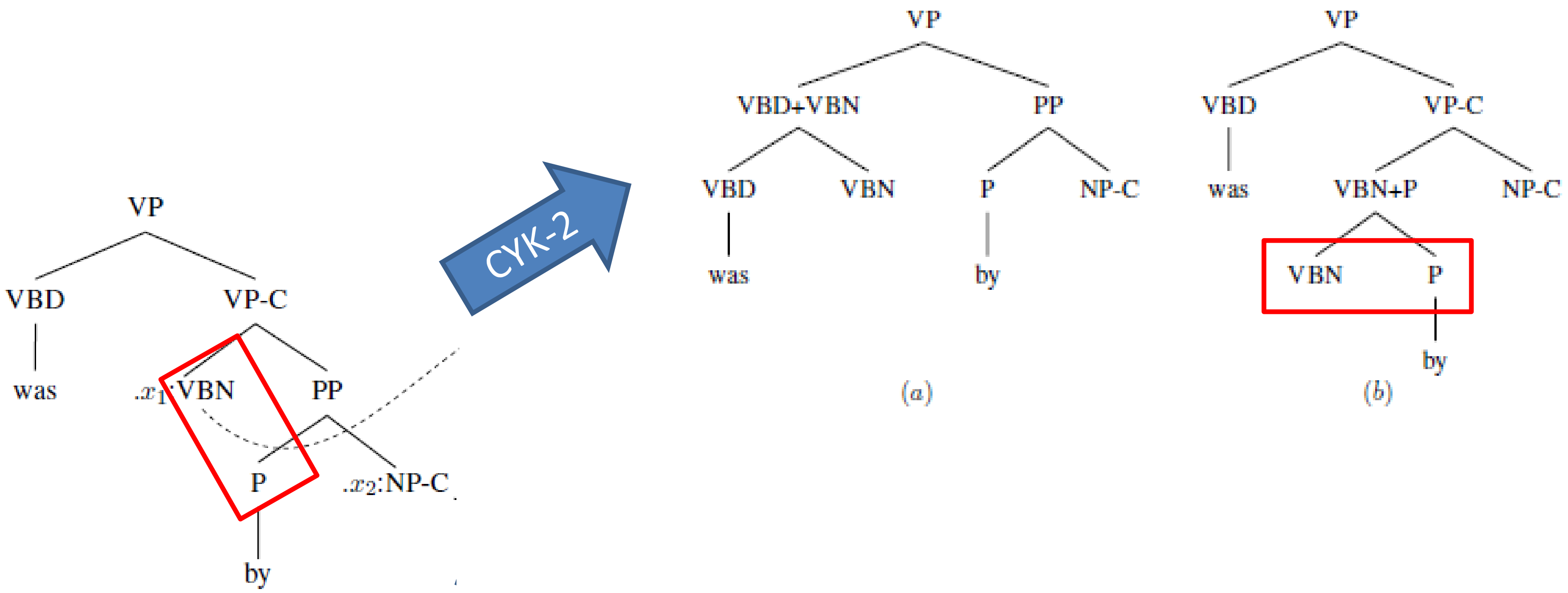
CYK-2 forest

the **frogs** **ate** **soup**

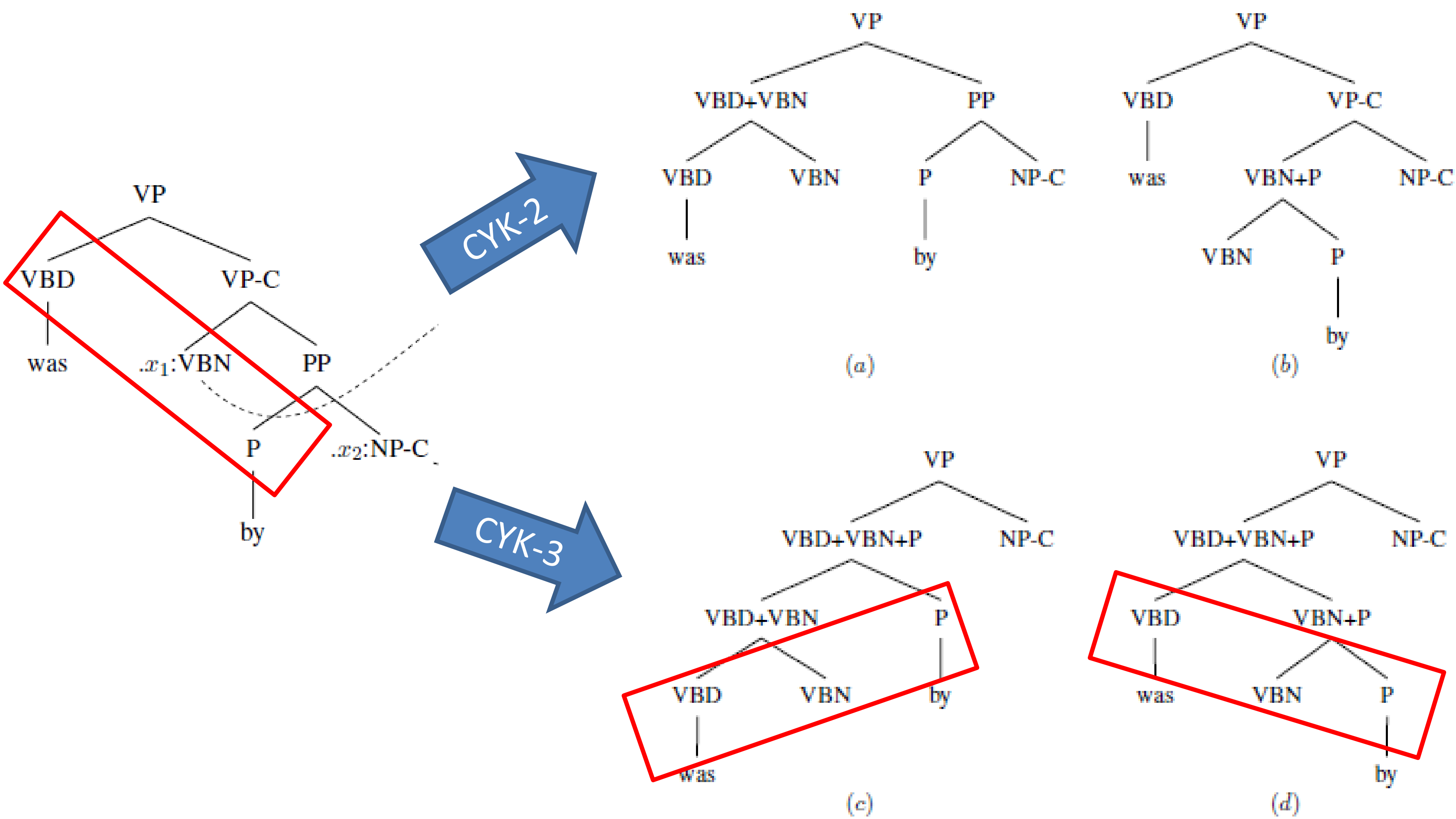
parse-forest construction



parse-forest construction



parse-forest construction



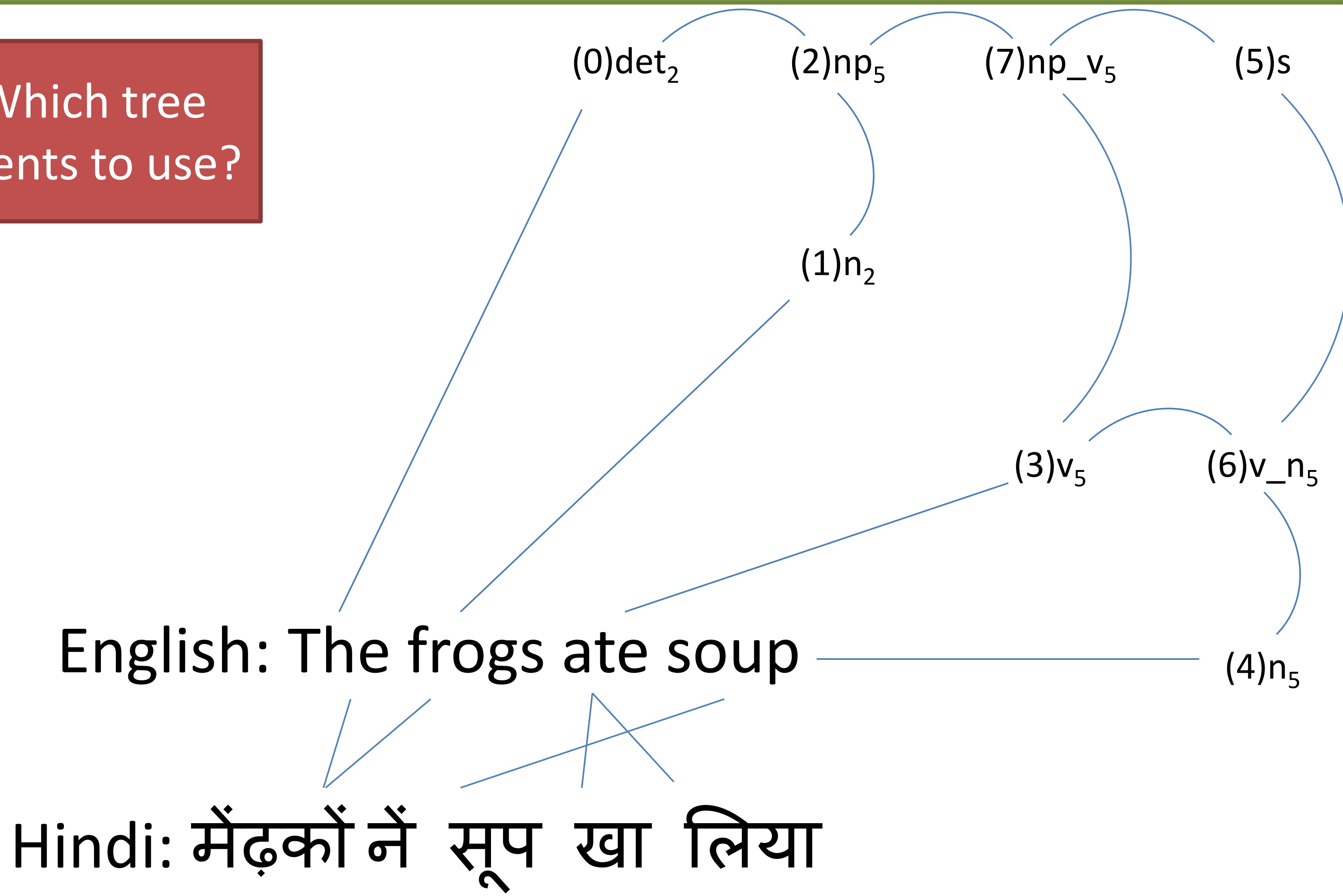
parse-forest construction

rule extraction

rule binarization

rule extraction

i.e. Which tree fragments to use?



synchronous tree substitution grammar

Phrase based rule: {“green houses”,
“maisons vertes”}

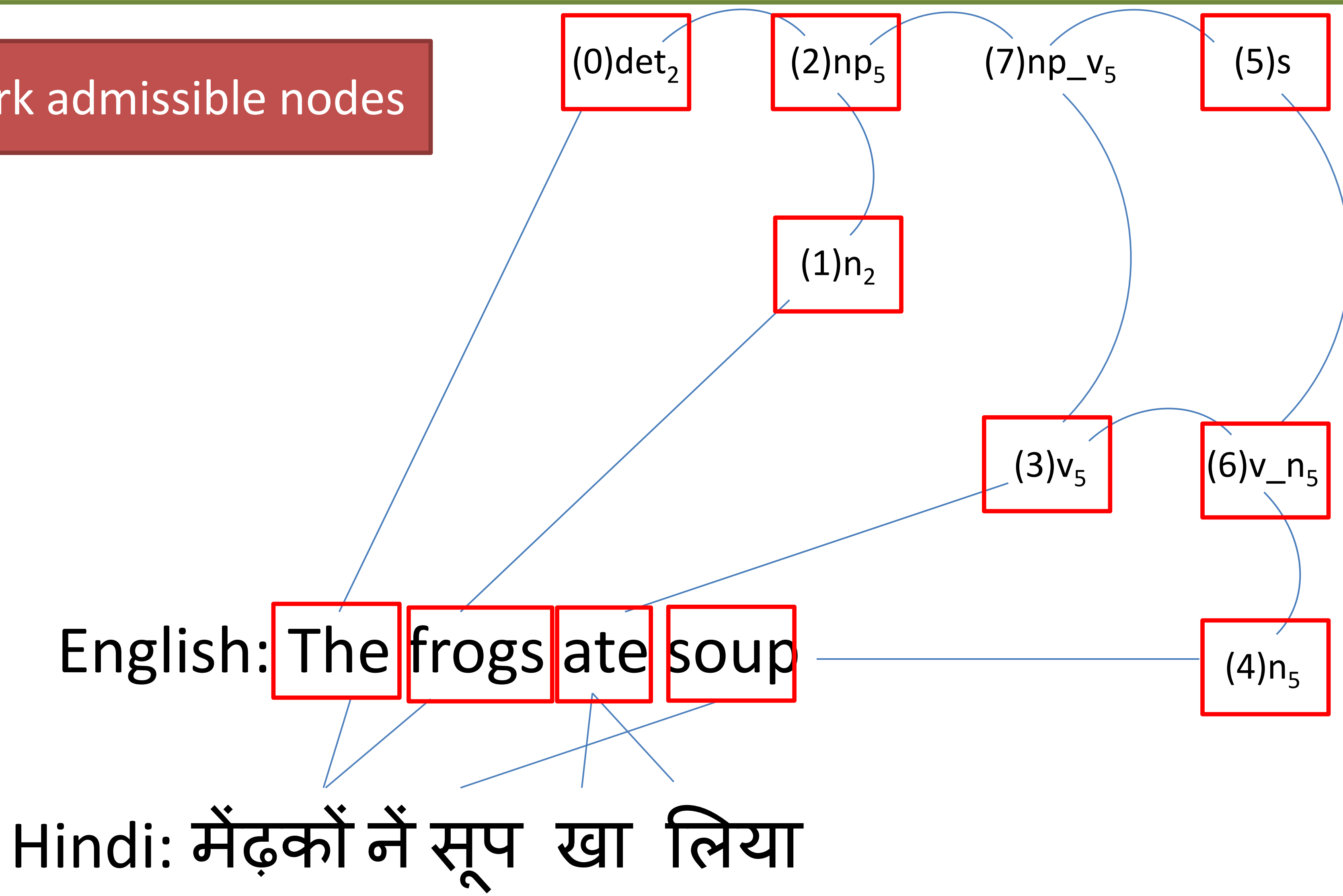
STSG rule1: {np(adj(green) n(house)),
np(n(maisons) adj(vertes))}

STSG rule2: {np(adj$_0$ n$_1$), np(n$_1$ adj$_0$)}

STSG rule3: {np(adj$_0$ n$_1$), “{1} {0}”}

rule extraction

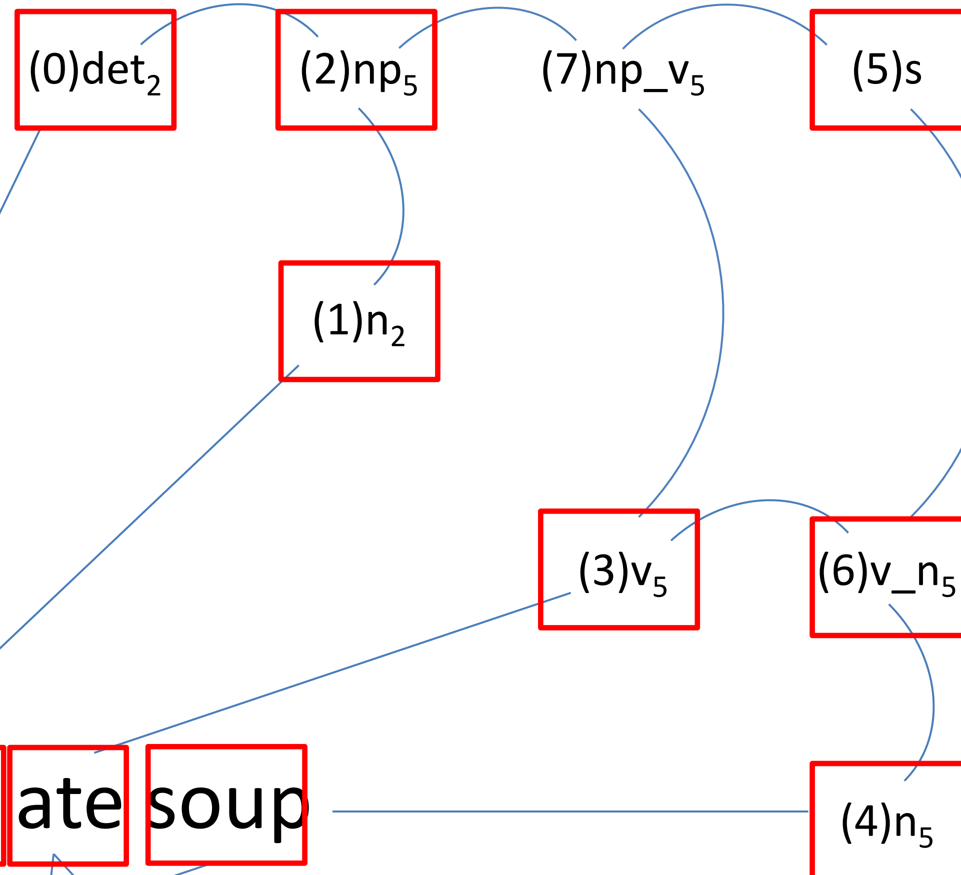
1. Mark admissible nodes



rule extraction

Seed fragment	Src span
the	0-0

2. Build tree fragments



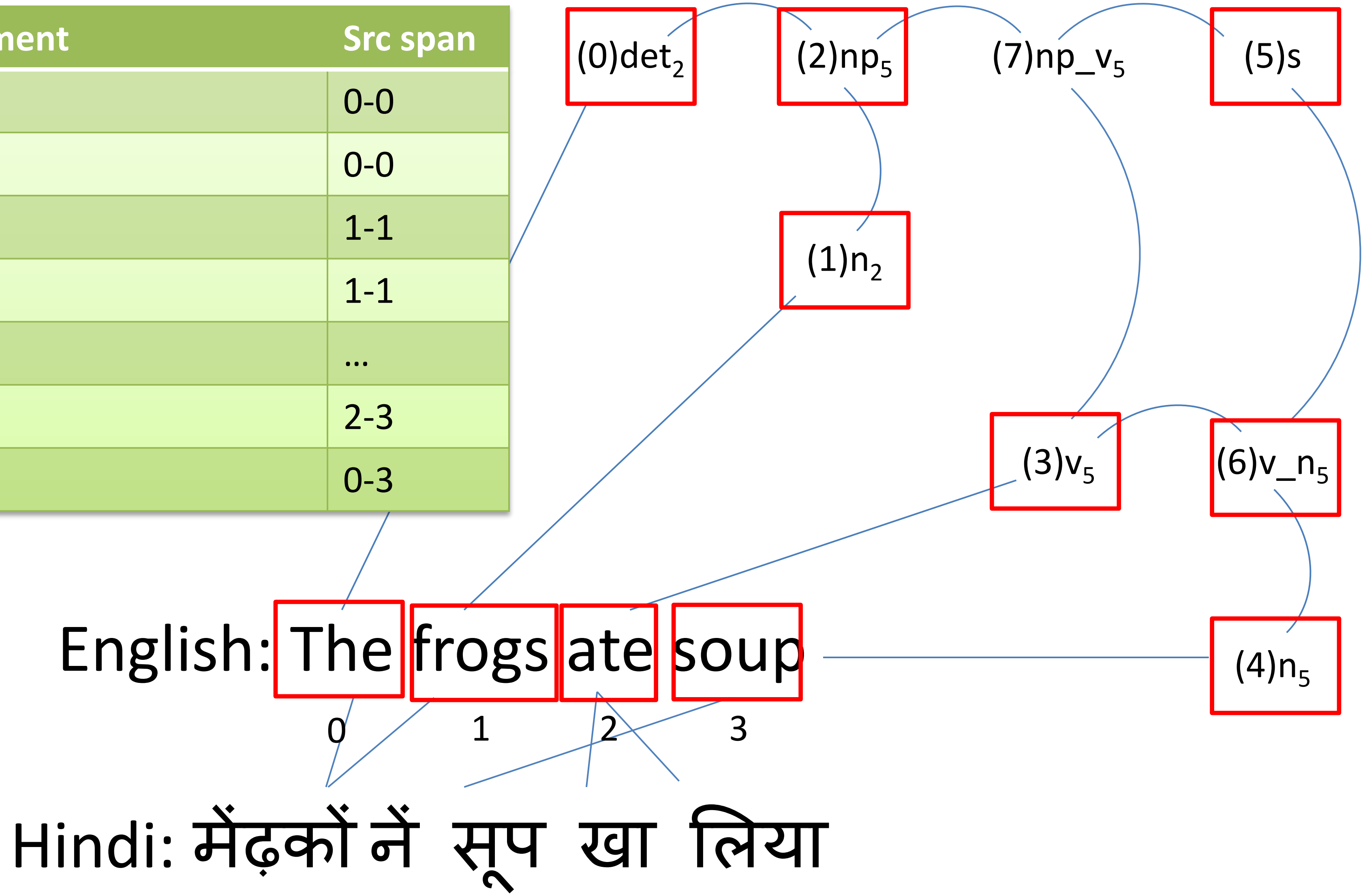
English: The frogs ate soup

0 1 2 3

Hindi: मेंढ़कों नें सूप खा लिया

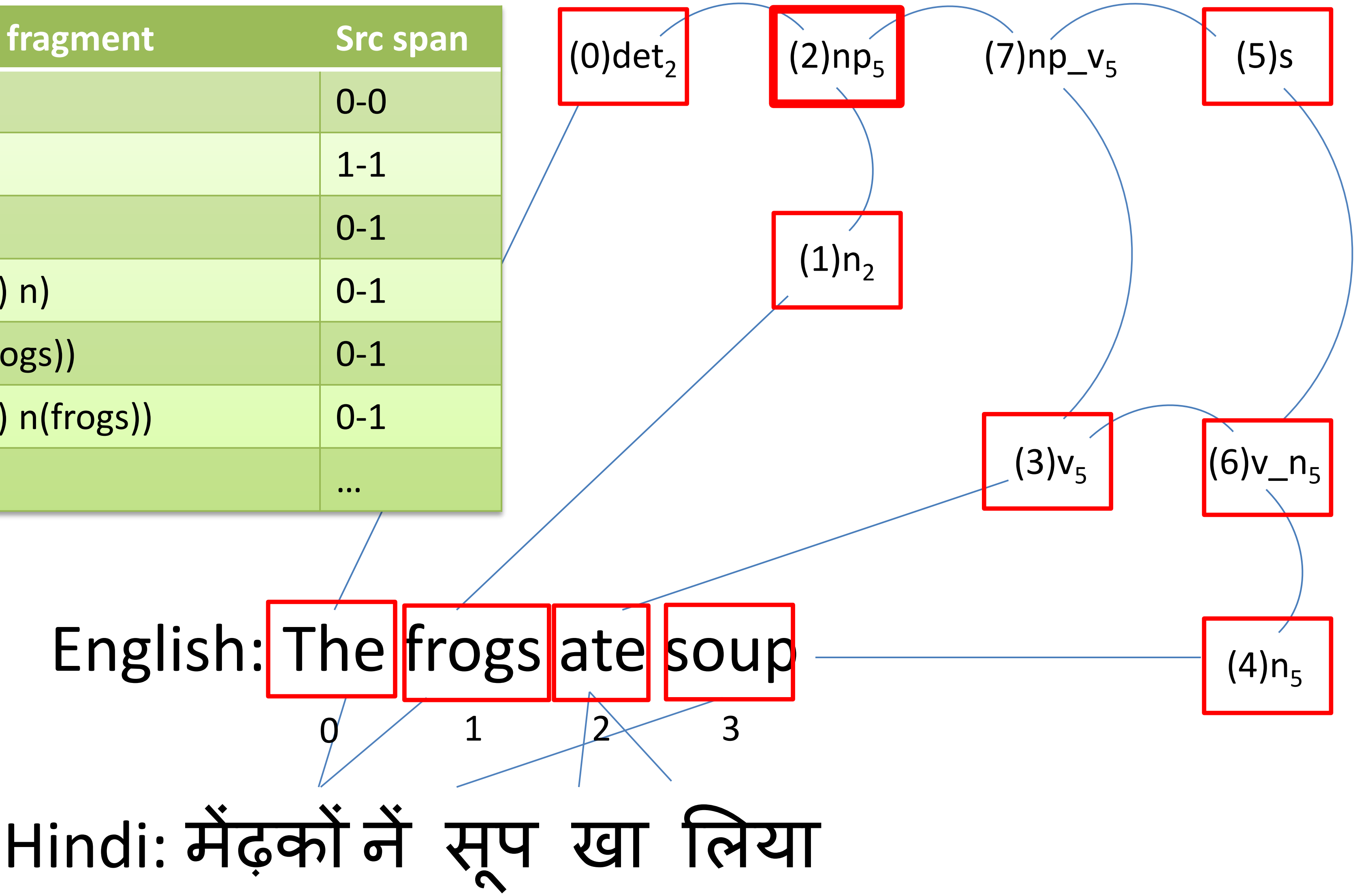
rule extraction

Seed fragment	Src span
The	0-0
det	0-0
frogs	1-1
n	1-1
…	…
v_n	2-3
s	0-3



rule extraction

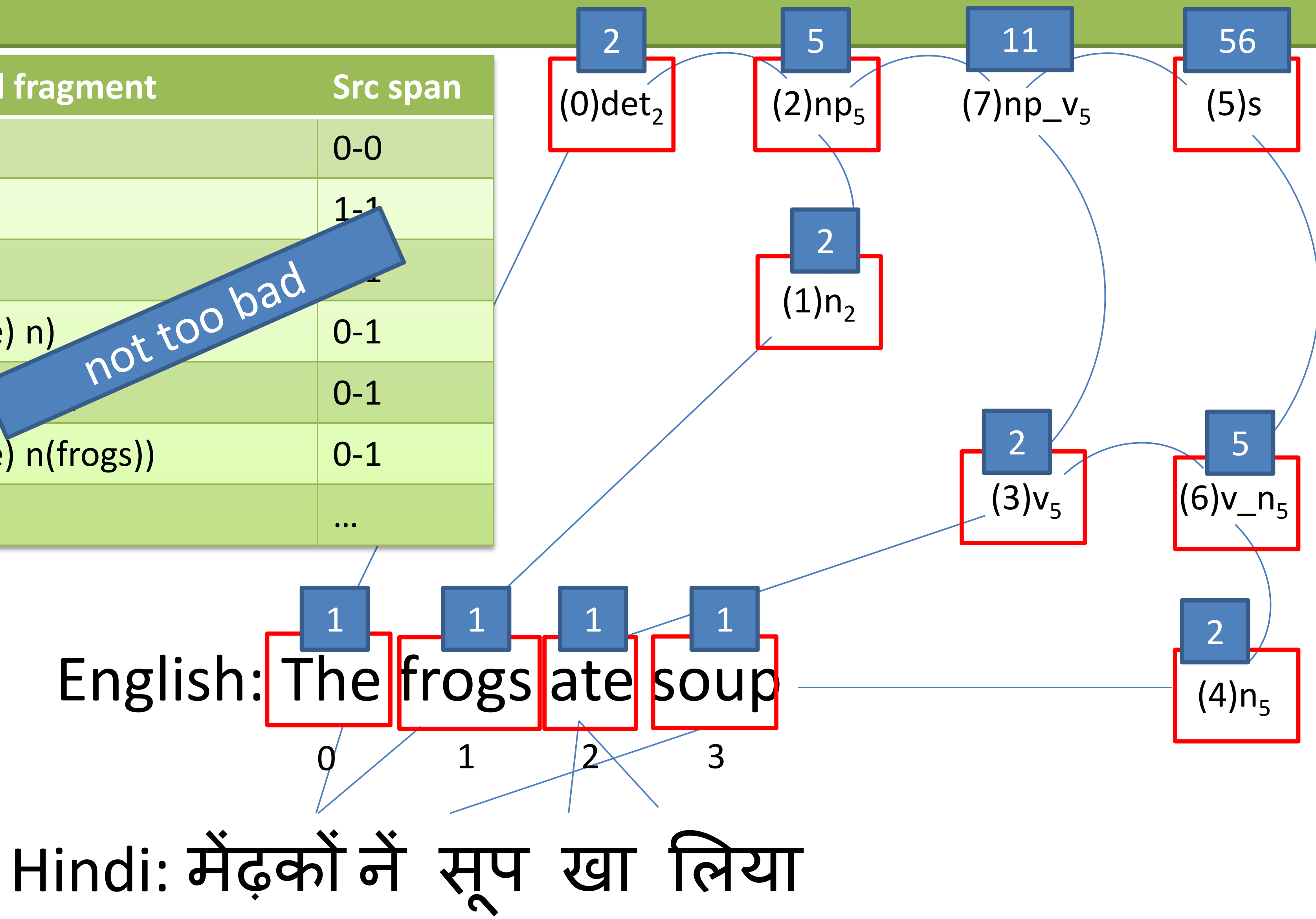
Composed fragment	Src span
det(the)	0-0
n(frogs)	1-1
np(det n)	0-1
np(det(the) n)	0-1
np(det n(frogs))	0-1
np(det(the) n(frogs))	0-1
…	…



rule extraction

Composed fragment	Src span
det(the)	0-0
n(frogs)	1-1
np(det n)	1
np(det(the) n)	0-1
np(det n(	0-1
np(det(the) n(frogs))	0-1
…	…

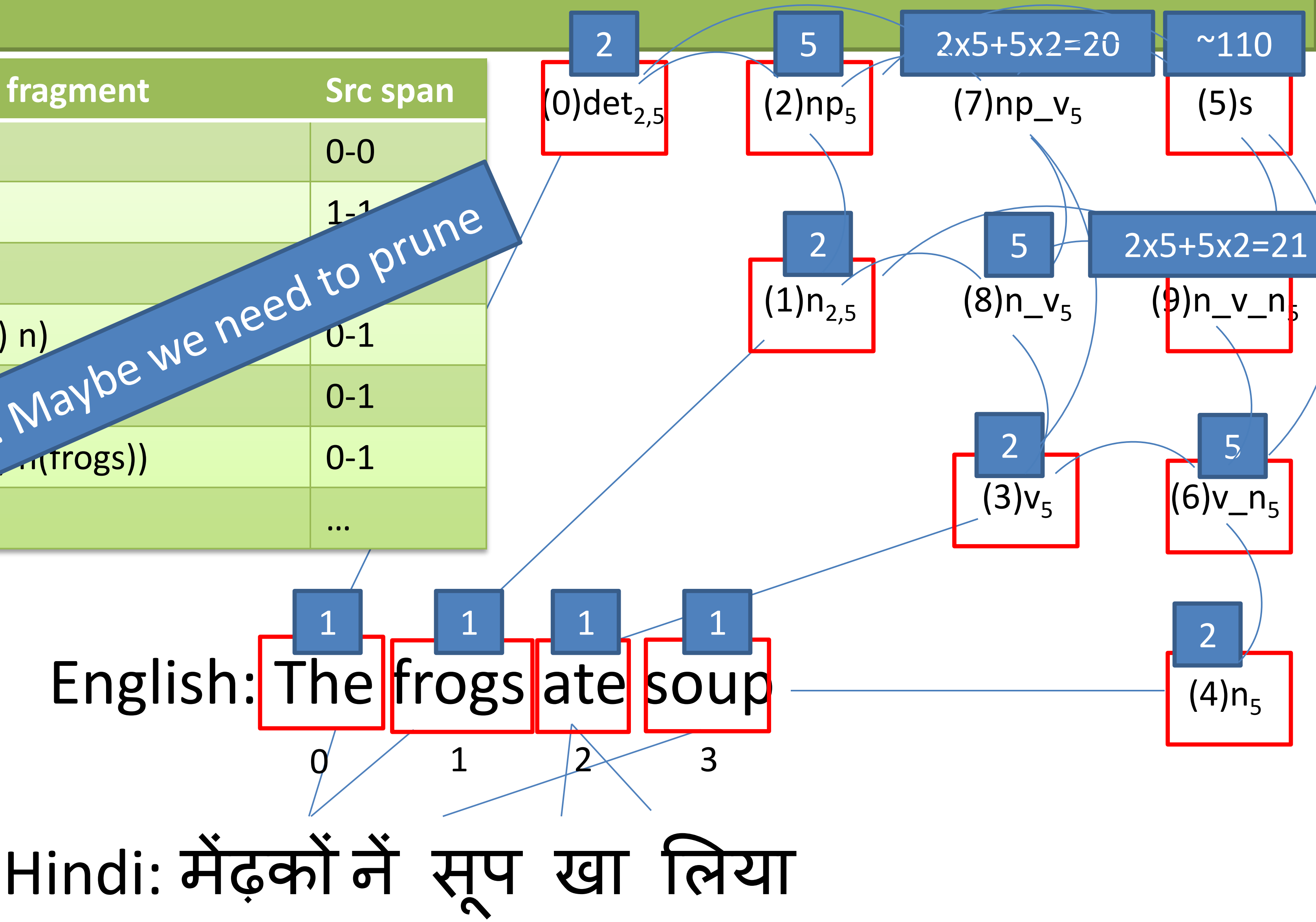
not too bad



rule extraction

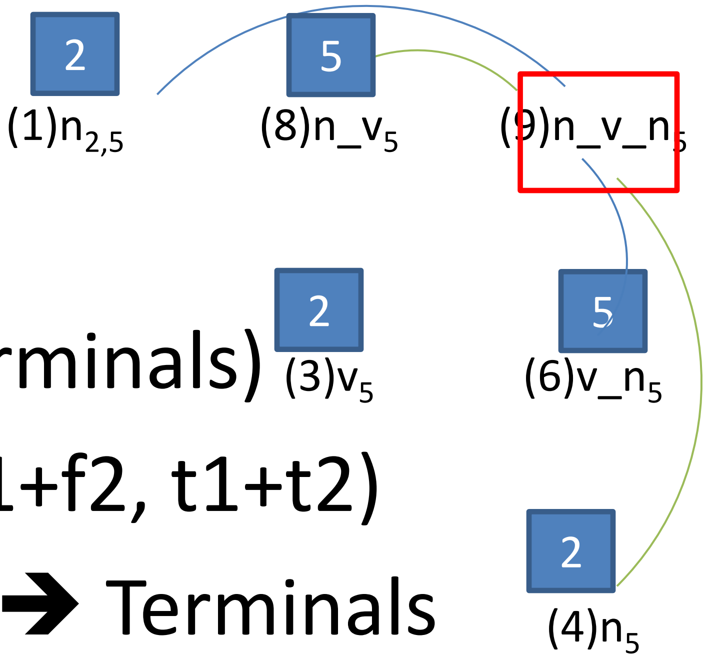
Composed fragment	Src span
det(the)	0-0
n(frogs)	1-1
np(det n)	
np(det(the) n)	0-1
np(det n(	0-1
n(frogs))	0-1
…	…

Hmm… Maybe we need to prune



rule extraction

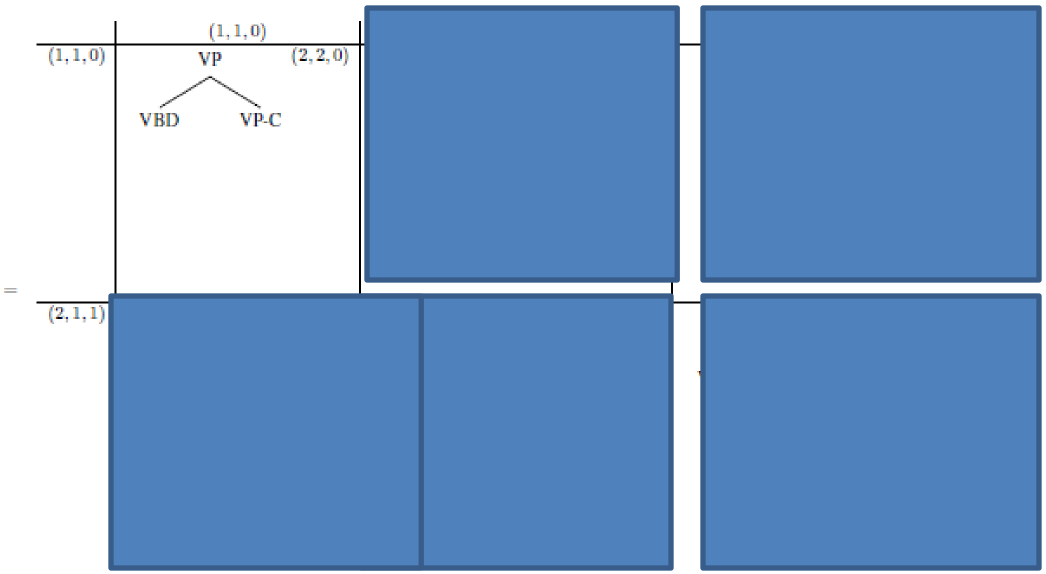
2. Build tree fragments



- Each node maintains a K-best list
- Score(rule) = (**h**eight,**f**rontiers,**t**erminals)
- Score(r1+r2) = (max{h1+h2}+1, f1+f2, t1+t2)
- Importance: Height ➔ Frontiers ➔ Terminals

rule extraction

2. Build tree fragments



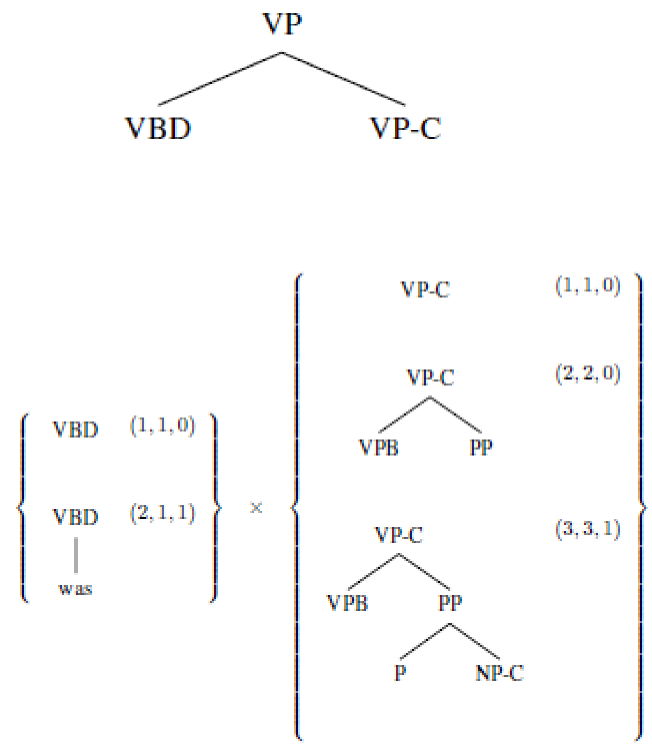
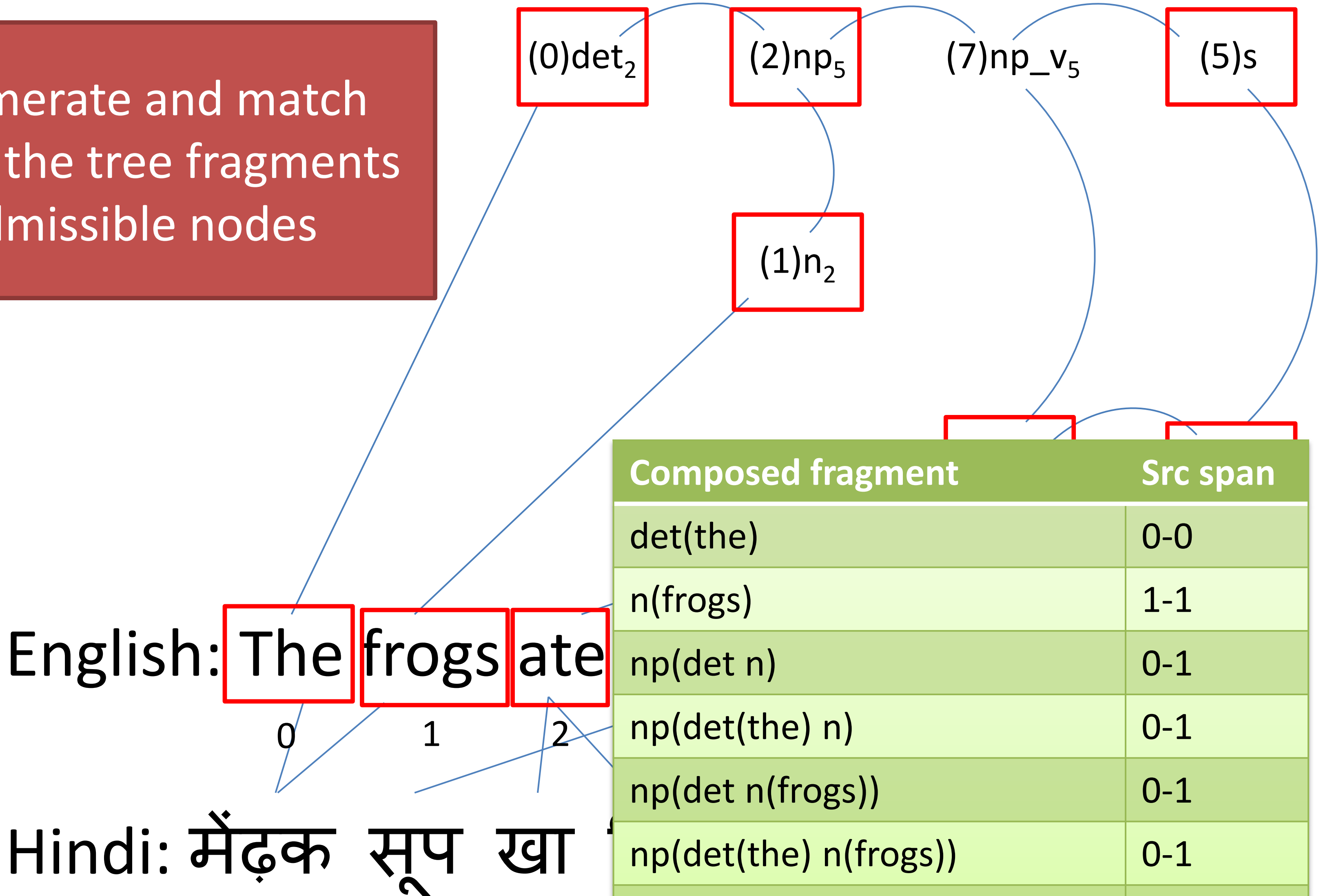


Figure 2: Tree-to-string rule composition as cube-pruning. The left shows two lists of composed rules sorted by their geometric measures ($height, \#\ frontiers, \#\ frontier\ terminals$), under the gluing rule of VP $\rightarrow$ VBD VP$-$C. The right part shows a cube view of the combination space. We explore the space from the top-left corner to the neighbors.

rule extraction

3. Enumerate and match rules for the tree fragments at admissible nodes



Composed fragment	Src span
det(the)	0-0
n(frogs)	1-1
np(det n)	0-1
np(det(the) n)	0-1
np(det n(frogs))	0-1
np(det(the) n(frogs))	0-1
…	…

parse-forest construction

rule extraction

$\boxed{\text{rule binarization}}$

rule binarization

- Rules are not necessarily binary

- More sharing
- Cube pruning

- Runtime vs. offline binarization

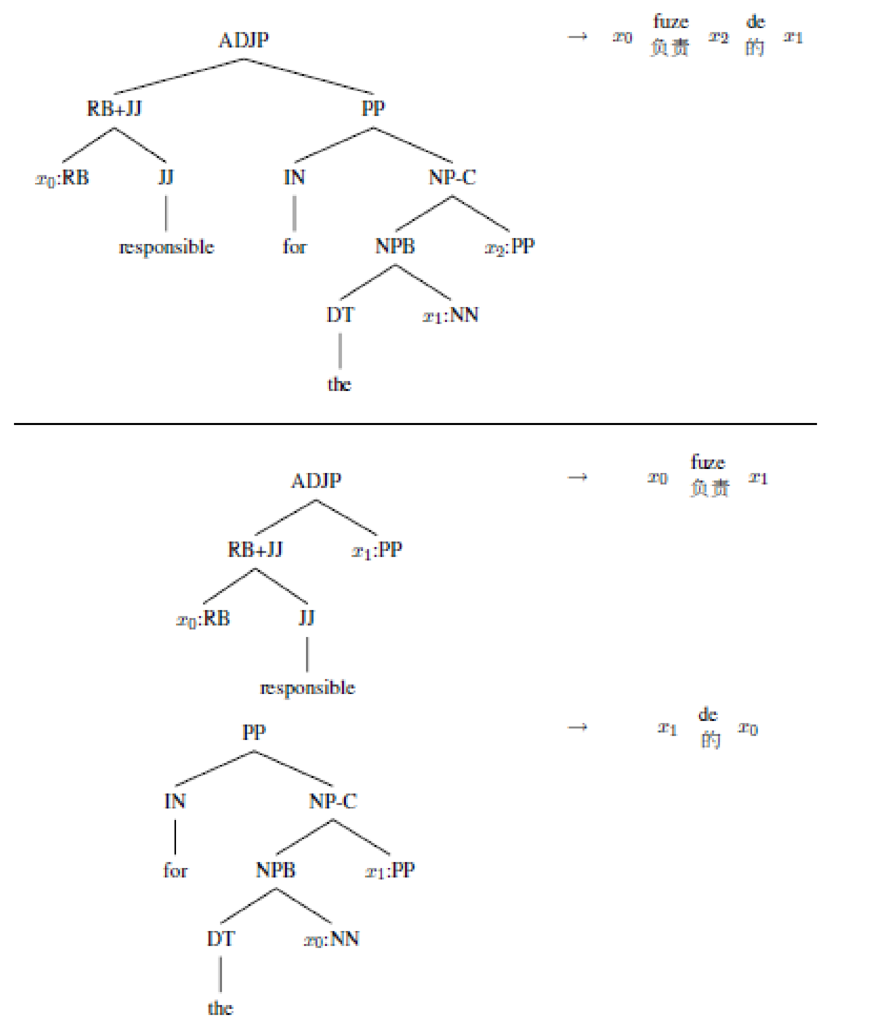


Figure 4: Synchronous binarization for a tree-to-string rule. The top rule can be binarized into two smaller rules.

experiments

- Compare to phrase-based and hiero
- 16 rules per node
- Shift-reduce dep. parser (87.8% labeled, 88.8% u)
- Map dependency trees to constituent trees

experiments

	Source Words	Target Words
English-Chinese	287M	254M
English-Czech	66M	57M
English-French	857M	996M
English-German	45M	43M
English-Spanish	216M	238M

Table 1: The Sizes of Parallel Texts.

	rules	BLEU dev	BLEU test
no binarization	378M	28.0	36.3
head-out	408M	30.0	38.2
cyk-1	527M	31.6	40.5
cyk-2	803M	31.9	40.7
cyk-3	1053M	32.0	40.6
cyk-∞	1441M	32.0	40.3

Table 3: Comparing different source tree binarization schemes for English-Chinese translation, showing both BLEU scores and model sizes. The rule counts include normal phrases which are used at the leaf level during decoding.

		BLEU dev	BLEU test
English-Chinese	pb	29.7	39.4
	hier	31.7	38.9
	bf2s	31.9	40.7**
English-Czech	*wmt best*	-	*15.4*
	pb	14.3	15.5
	hier	14.7	16.0
	bf2s	14.8	16.3*
English-French	*wmt best*	-	*27.6*
	pb	24.1	26.1
	hier	23.9	26.1
	bf2s	24.5	26.6**
English-German	*wmt best*	-	*16.3*
	pb	14.5	15.5
	hier	14.9	15.9
	bf2s	15.2	16.3**
English-Spanish	*wmt best*	-	*28.4*
	pb	24.1	27.9
	hier	24.2	28.4
	bf2s	24.9	28.9**

Table 2: Translation results comparing *bf2s*, the binarized-forest-to-string system, *pb*, the phrase-based system, and *hier*, the hierarchical phrase-based system.

experiments

- Optimal degree for CYK-n

	rules	BLEU dev	BLEU test
no binarization	378M	28.0	36.3
head-out	408M	30.0	38.2
cyk-1	527M	31.6	40.5
cyk-2	803M	31.9	40.7
cyk-3	1053M	32.0	40.6
cyk-∞	1441M	32.0	40.3

Table 3: Comparing different source tree binarization schemes for English-Chinese translation, showing both BLEU scores and model sizes. The rule counts include normal phrases which are used at the leaf level during decoding.

experiments

- Binarized forests vs. parse-generated forests

	BLEU	
	dev	test
cyk-2	14.9	16.0
parser	14.7	15.7

Table 4: Binarized forests versus parser-generated forests for forest-to-string English-German translation.

experiments

- Effect of rule binarization

		BLEU	
		dev	test
head-out	cube pruning	29.2	37.0
	+ synch. binarization	30.0	38.2
cyk-2	cube pruning	31.7	40.5
	+ synch. binarization	31.9	40.7

Table 5: The effect of synchronous binarization for tree-to-string and forest-to-string systems, on the English-Chinese task.

discussion