Advanced PCFG models

Computational Linguistics

Alexander Koller

4 December 2018

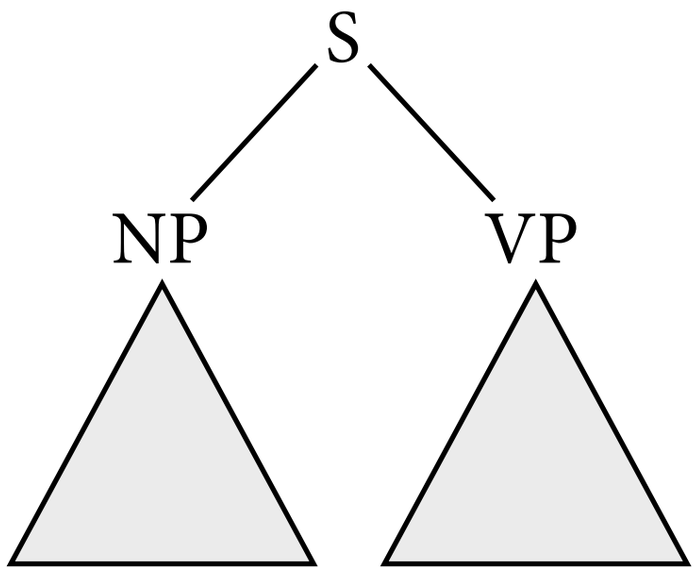
The story so far

- Train PCFG with MLE on the Penn Treebank 02-21.
- Compute parse trees for PTB 23 using Viterbi-CKY.
- Trick against data sparseness in lexicon:
  delete words, train and test on sequences of POS tags.
- This yields labeled f-score in the low 70's.
  - Why so low?
  - How can we fix it?

Fundamental problem of PCFGs

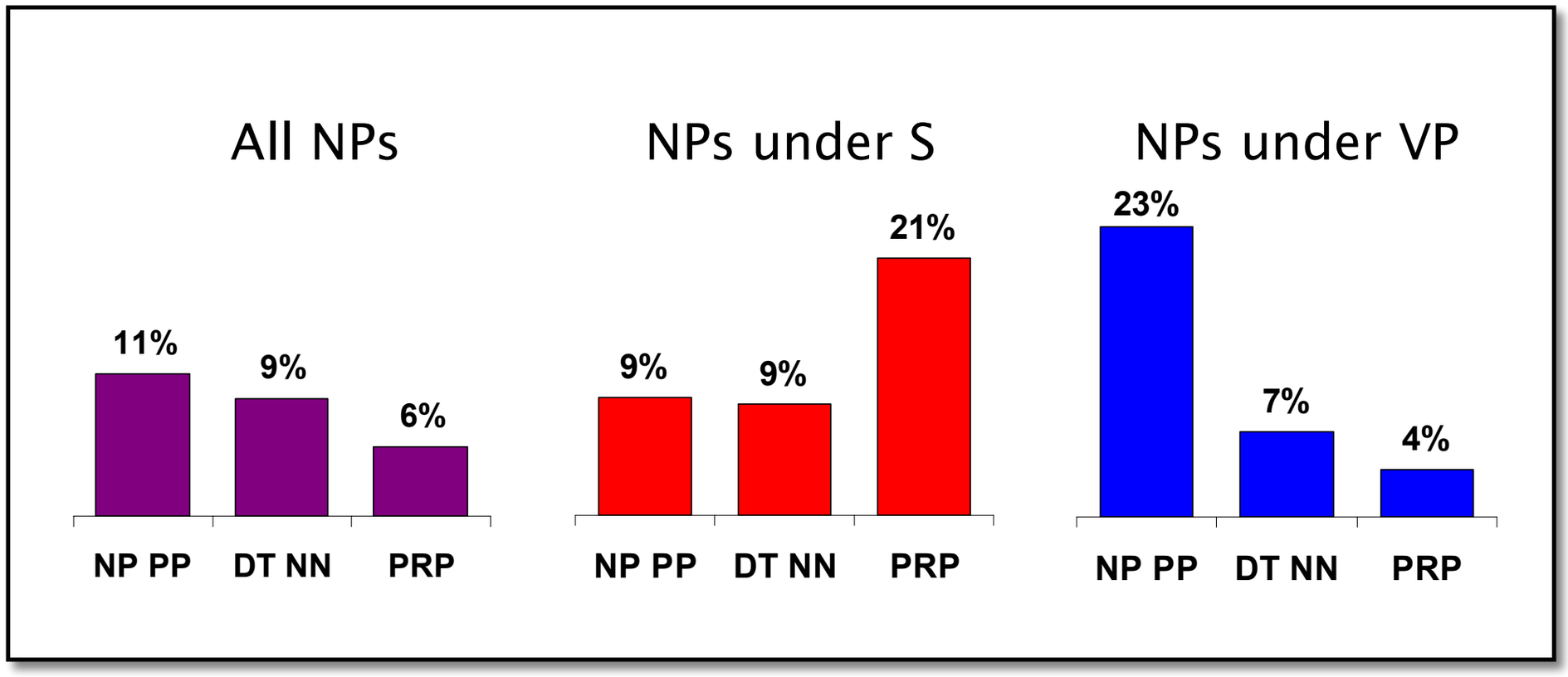
- Context-free grammar: One rule can only "see" parent and its children, not anything above or below.

- PCFG: Assumes statistical independence of all rewrite events.



NP → PRP?
NP → Det N?

Independence assumptions



PTB statistics, from slides by Dan Klein

Independence assumptions

- Accurate disambiguation of PP attachment requires lexical information.
  - I shot the elephant with a long trunk.
  - I shot the elephant with a long rifle.

- PP attachment influenced by choice of P.
  - Collins note: “workers dumped sacks *into* a bin”
  - *into*-PPs in PTB 9x more likely to attach to VP than to N

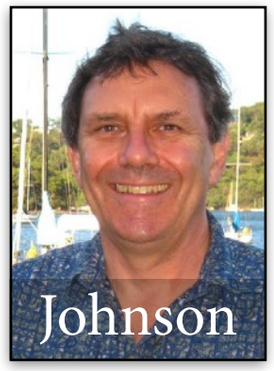
- PCFGs rely on nonterminals alone,
  cannot “see” lexical information.

Directions

- Need to make nonterminals more informative to make PCFG rules sensitive to more context.

- Several approaches discussed today:
  - Johnson 98: Parent annotations
  - Collins 97: Lexicalized PCFGs
  - Klein & Manning 03: Unlexicalized PCFGs with nonterminals split by hand
  - Petrov & Klein 06: Unlexicalized PCFGs with automatically learned nonterminal splits

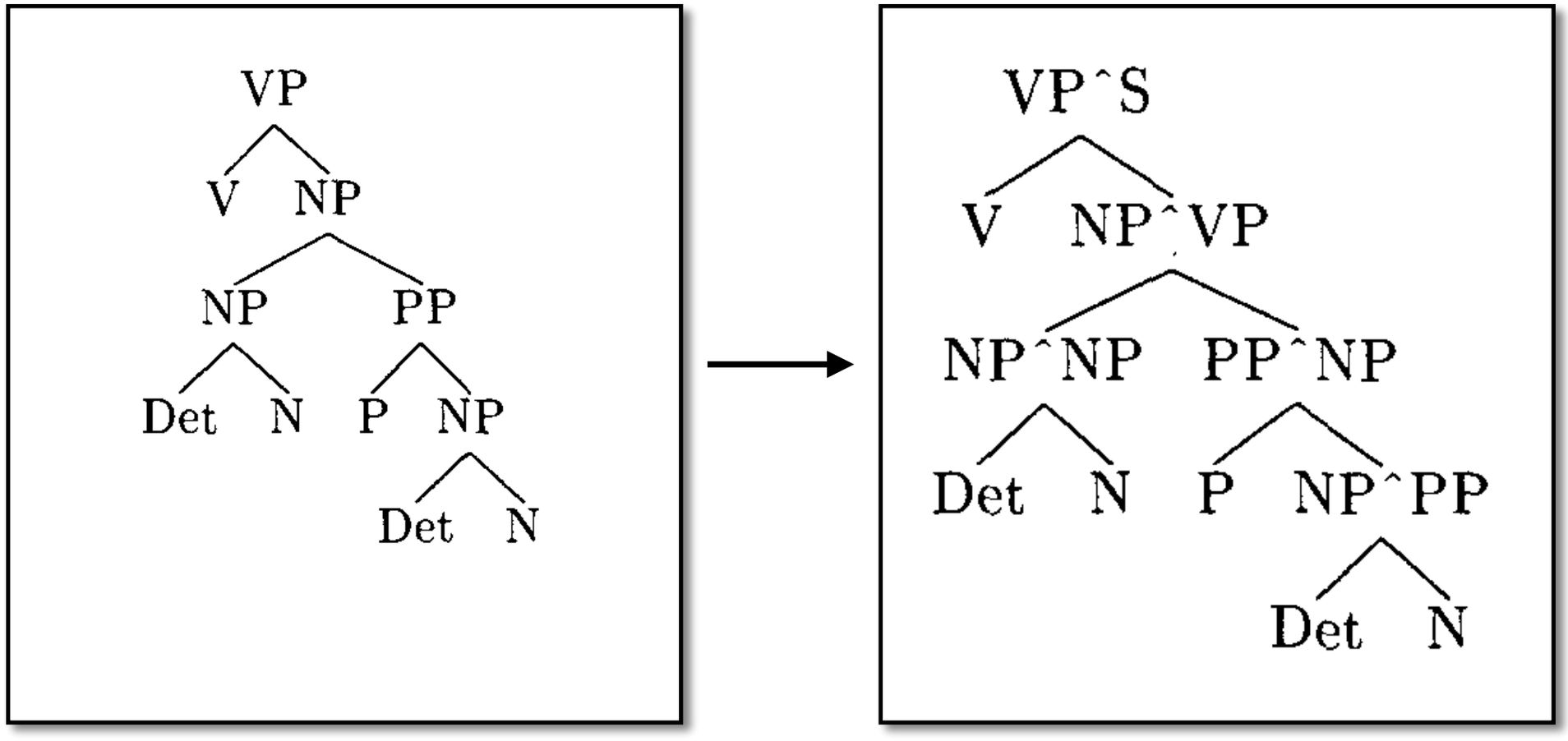
Johnson 1998



- Discusses PTB preprocessing and impact of PTB representation changes.

- One key idea: *parent annotations*.
  - If parent of NP makes such a difference in how it should be expanded, why don't we encode the parent of the NP?
  - Replace nonterminal NP by NP^S (NP as child of S), NP^VP (NP as child of VP), and so on in PTB trees.
  - Train grammar on modified treebank.
    After parsing, remove annotations and compare to gold standard tree.

Example

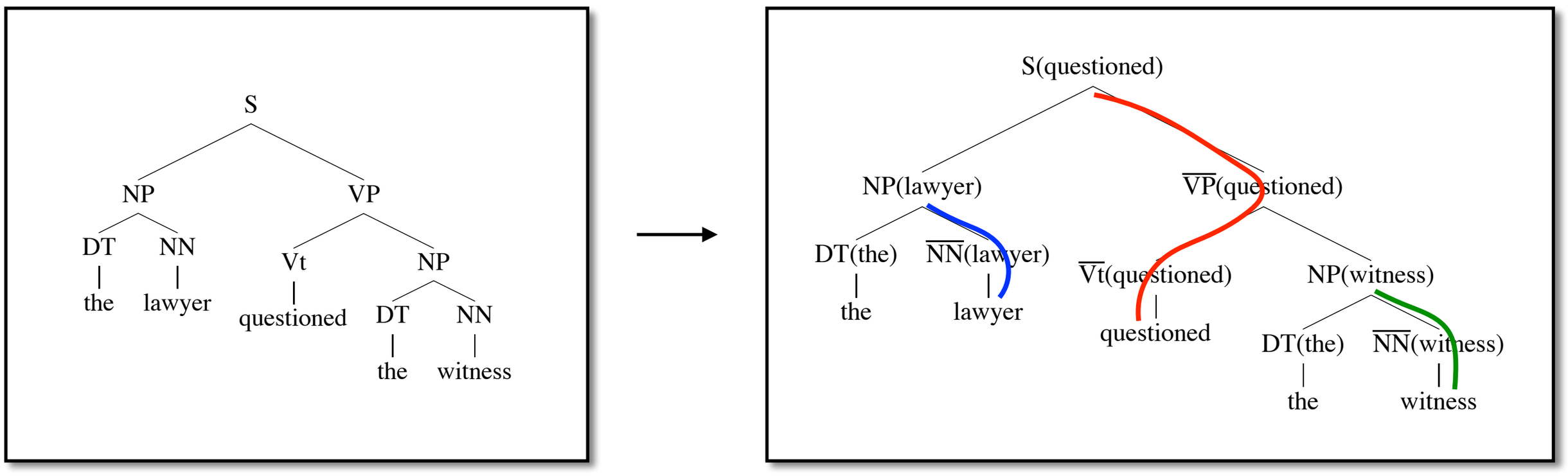


Result: Labeled f-score on Section 22 jumps from 71.5 to 79.6.
Number of production rules grows from 15,000 to 22,000.

Lexicalized parsing

- Fundamental idea: If words are so important to distribution of rules, let's put them in the rules.

- Step 1: Mark each node in PTB with its *lexical head*.
  - identify head automatically using hand-written rules



Lexicalized PCFGs

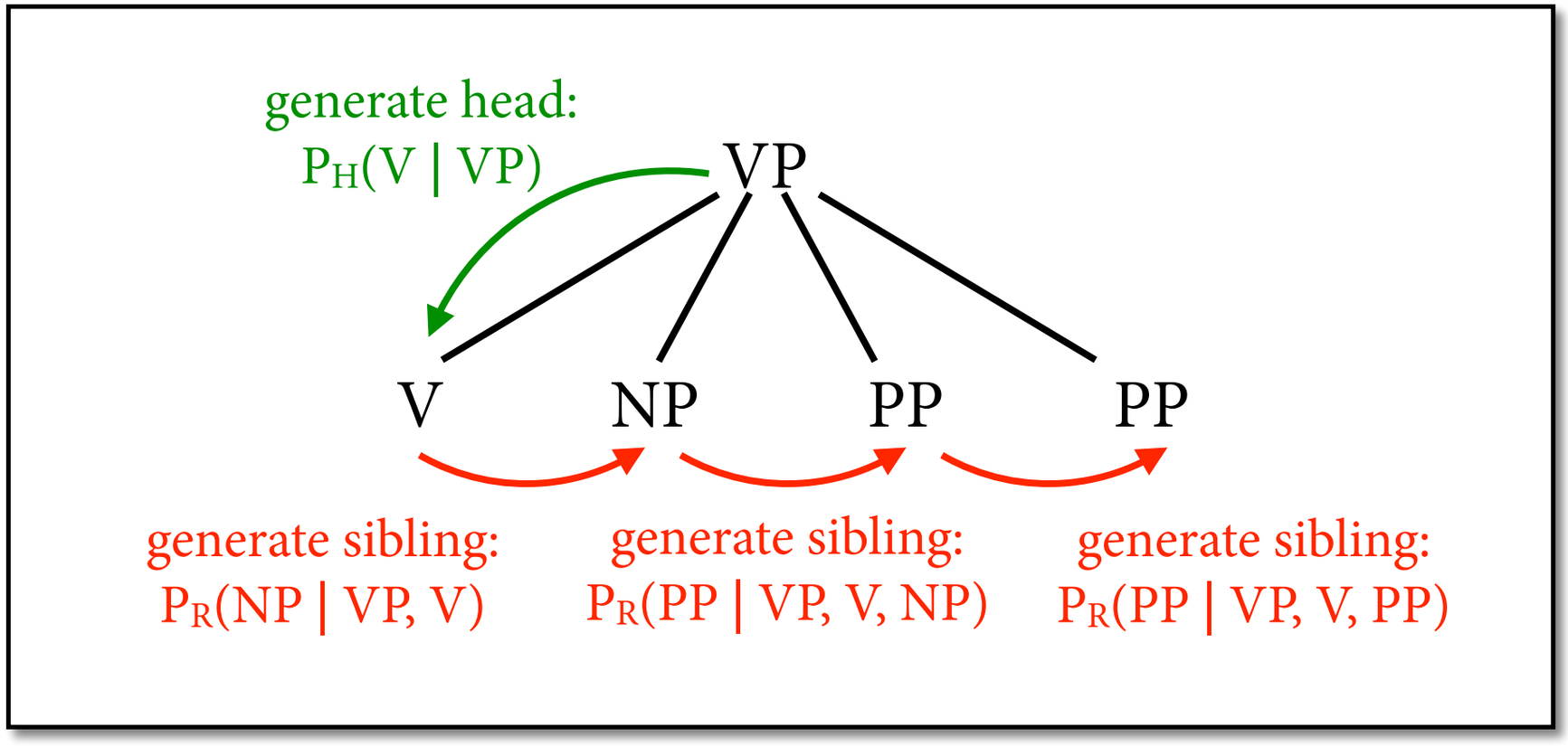
- Step 2: Read off lexicalized PCFG from treebank.
  - rules of the form S(examined) $\rightarrow_2$ NP(lawyer) VP(examined)
  - "2" on arrow indicates that second child is head

- MLE and Viterbi-CKY adapt easily to new setting.
  So we're basically done!

- But! Number of rules multiplied by V^r
  (V = vocabulary size, r = rank of rules).
  - ordinary rule × head word × heads of other children
  - increases number of parameters accordingly
  - astronomical sparse data problem

Dealing with sparse data

- Horizontal Markovization:
  - break rules up into parts by generating children one by one
  - independence assumptions: child depends on limited context



Dealing with sparse data

- This helps a lot, but is still not enough for rare events.

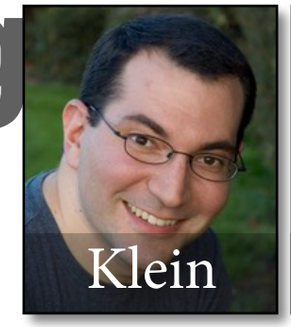
- Need aggressive smoothing. Collins uses interpolation:
  - p_1 = C(S → NP VP, H = examined) / C(S, H = examined)
  - p_0 = C(S → NP VP) / C(S)
  - P(S → NP VP | S, examined) = $\lambda\ p_1 + (1-\lambda)\ p_0$
  - estimate λ from data

Collins 1997 (with more complex lexicalization model): f-score 87.7 on PTB word strings of length ≤ 40

Parsing speed

- Parsing slower than usual, because
  - grammar is much bigger
  - must be careful in managing head words
- Key insight: head word of (A,i,k) must be one of w_i, …, w_{k-1}; use pointers into input string.
  - this gives $O(n^5)$ parsing time with acceptable memory use
  - Eisner & Satta 99: can do it in $O(n^4)$ with clever algorithm — still too slow in practice
  - use beam search to maintain only best hypotheses for each chart cell

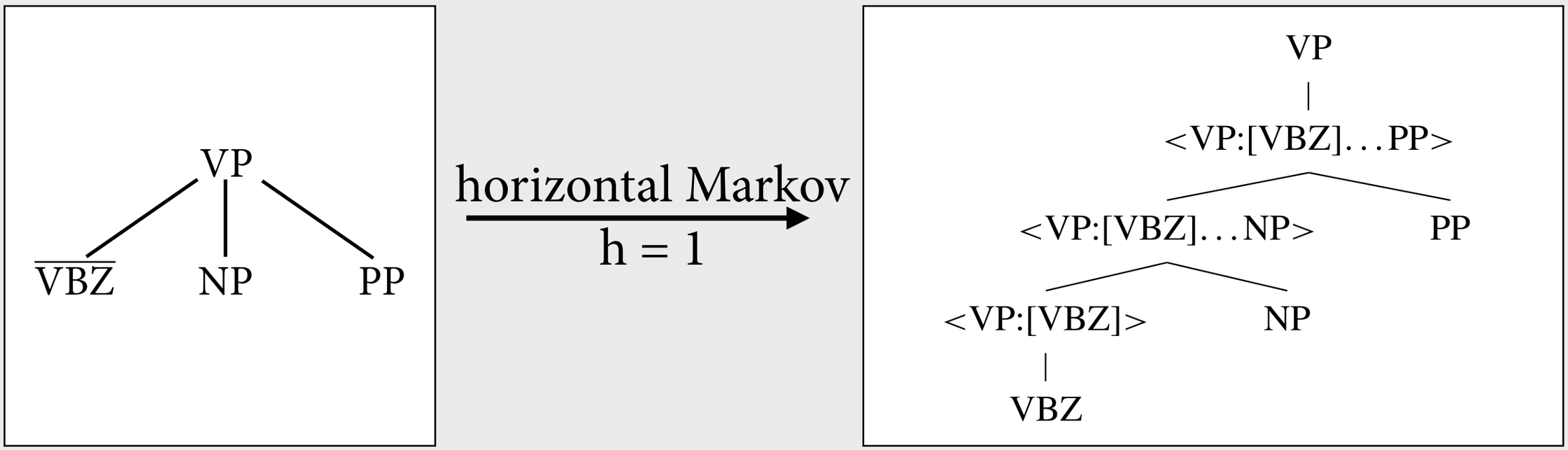
Unlexicalized parsing



- Is lexicalization really as helpful as it seems?
  - Gildea 01: what counts is effect of head word on choice of subcategorization frame, not bilexical dependencies
  - Dubey & Keller 03: bilexical dependencies not useful when parsing German
  - Even lexicalized parsers (e.g. Collins 99, Charniak 00) make use of non-lexical splits of nonterminals.

- Klein & Manning 03: Perhaps usefulness of lexicalization is primarily in giving us more nonterminals?
  Can we get the same effect more cheaply?

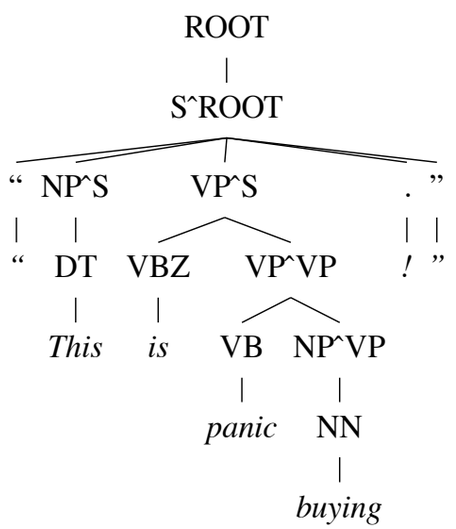
Markovization

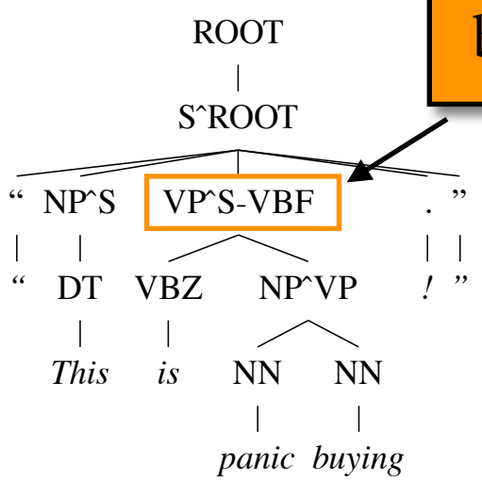


Vertical Markovization:
v = 2 is parent annotations
v = 3 grandparent, etc.

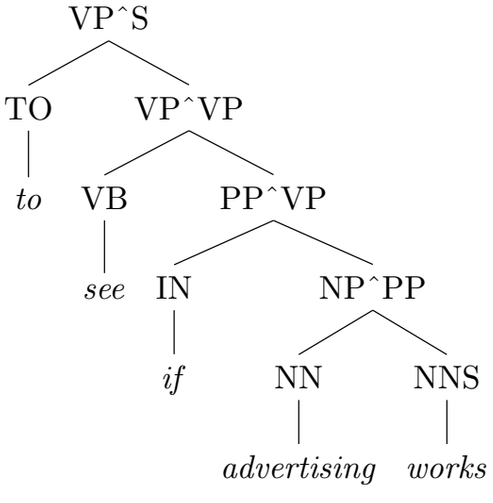
Vertical Order		Horizontal Markov Order $h=0$	$h=1$	$h\leq 2$	$h=2$	$h=\infty$
$v=1$	No annotation	71.27 (854)	72.5 (3119)	73.46 (3863)	72.96 (6207)	72.62 (9657)
$v\leq 2$	Sel. Parents	74.75 (2285)	77.42 (6564)	77.77 (7619)	77.50 (11398)	76.91 (14247)
$v=2$	All Parents	74.68 (2984)	77.42 (7312)	77.81 (8367)	77.50 (12132)	76.81 (14666)
$v\leq 3$	Sel. GParents	76.50 (4943)	78.59 (12374)	79.07 (13627)	78.97 (19545)	78.54 (20123)
$v=3$	All GParents	76.74 (7797)	79.18 (15740)	79.74 (16994)	79.07 (22886)	78.72 (22002)

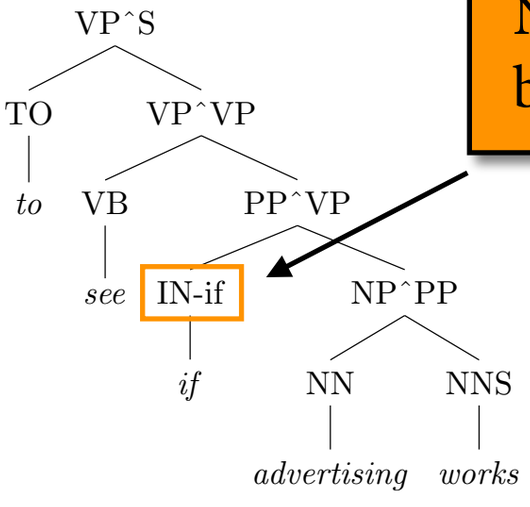
Rule-based state splitting





Not just an VP^S,
but one whose head is a finite verb.





Not just a preposition,
but one that is like “if”.

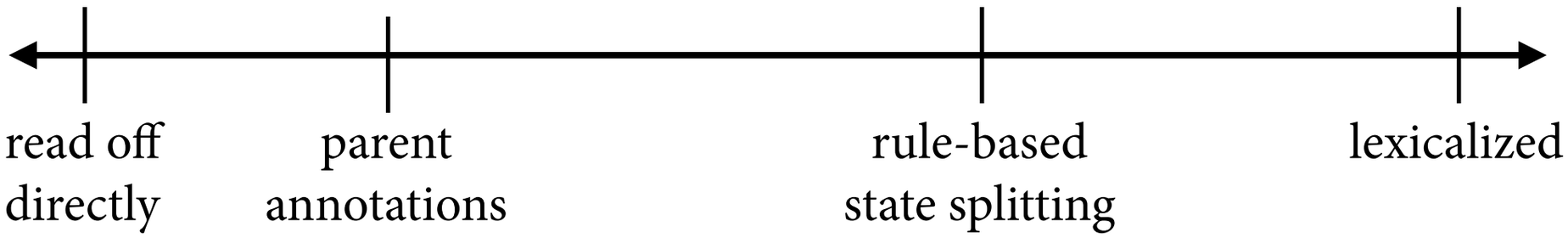
Results

	Cumulative			Indiv.
Annotation	Size	F_1	ΔF_1	ΔF_1
Baseline ($v \leq 2, h \leq 2$)	7619	77.77	–	–
UNARY-INTERNAL	8065	78.32	0.55	0.55
UNARY-DT	8066	78.48	0.71	0.17
UNARY-RB	8069	78.86	1.09	0.43
TAG-PA	8520	80.62	2.85	2.52
SPLIT-IN	8541	81.19	3.42	2.12
SPLIT-AUX	9034	81.66	3.89	0.57
SPLIT-CC	9190	81.69	3.92	0.12
SPLIT-%	9255	81.81	4.04	0.15
TMP-NP	9594	82.25	4.48	1.07
GAPPED-S	9741	82.28	4.51	0.17
POSS-NP	9820	83.06	5.29	0.28
SPLIT-VP	10499	85.72	7.95	1.36
BASE-NP	11660	86.04	8.27	0.73
DOMINATES-V	14097	86.91	9.14	1.42
RIGHT-REC-NP	15276	87.04	9.27	1.94

Compare against f-score 87-89 of lexicalized parsers.
But much smaller grammars, simpler and faster parsing!

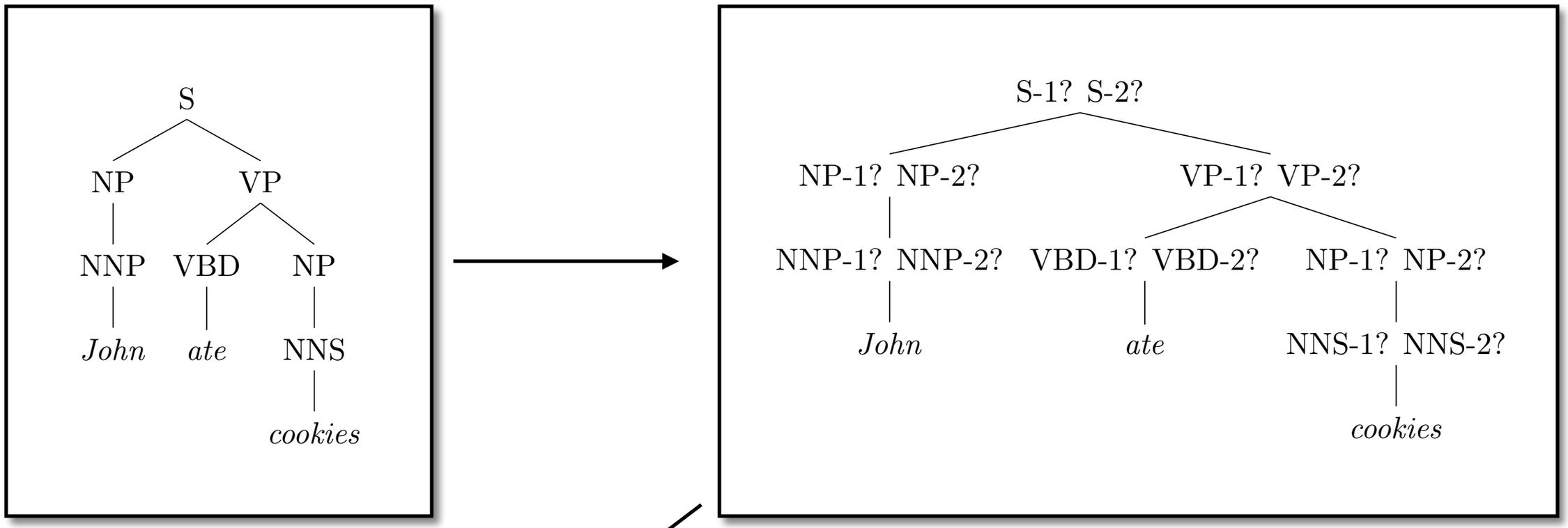
State splitting

- Can see all of these approaches as methods for refining the nonterminals of the PTB.



- Petrov et al. 06: Can we automatically learn how to refine ("split") the nonterminals?

Split-Merge



original annotation

possible trees with state-split nonterminals

S-1 → NP-1 VP-1
S-1 → NP-1 VP-2
S-1 → NP-2 VP-1 etc.

grammar with state-split NTs

learn weights using EM

+ “merge”: undo useless splits

Results

$\leq$ 40 words	LP	LR	CB	0CB
Klein and Manning (2003)	86.9	85.7	1.10	60.3
Matsuzaki et al. (2005)	86.6	86.7	1.19	61.1
Collins (1999)	88.7	88.5	0.92	66.7
Charniak and Johnson (2005)	90.1	**90.1**	**0.74**	**70.1**
This Paper	**90.3**	90.0	0.78	68.5

all sentences	LP	LR	CB	0CB
Klein and Manning (2003)	86.3	85.1	1.31	57.2
Matsuzaki et al. (2005)	86.1	86.0	1.39	58.3
Collins (1999)	88.3	88.1	1.06	64.0
Charniak and Johnson (2005)	89.5	**89.6**	**0.88**	**67.6**
This Paper	**89.8**	**89.6**	0.92	66.3

(“this paper” = Petrov et al. 06)

Some state-split POS tags

VBZ

VBZ-0	gives	sells	takes
VBZ-1	comes	goes	works
VBZ-2	includes	owns	is
VBZ-3	puts	provides	takes
VBZ-4	says	adds	Says
VBZ-5	believes	means	thinks
VBZ-6	expects	makes	calls
VBZ-7	plans	expects	wants
VBZ-8	is	’s	gets
VBZ-9	’s	is	remains
VBZ-10	has	’s	is
VBZ-11	does	Is	Does

NNP

NNP-0	Jr.	Goldman	INC.
NNP-1	Bush	Noriega	Peters
NNP-2	J.	E.	L.
NNP-3	York	Francisco	Street
NNP-4	Inc	Exchange	Co
NNP-5	Inc.	Corp.	Co.
NNP-6	Stock	Exchange	York
NNP-7	Corp.	Inc.	Group
NNP-8	Congress	Japan	IBM
NNP-9	Friday	September	August
NNP-10	Shearson	D.	Ford
NNP-11	U.S.	Treasury	Senate
NNP-12	John	Robert	James
NNP-13	Mr.	Ms.	President
NNP-14	Oct.	Nov.	Sept.
NNP-15	New	San	Wall

JJS

JJS-0	largest	latest	biggest
JJS-1	least	best	worst
JJS-2	most	Most	least

DT

DT-0	the	The	a
DT-1	A	An	Another
DT-2	The	No	This
DT-3	The	Some	These
DT-4	all	those	some
DT-5	some	these	both
DT-6	That	This	each
DT-7	this	that	each
DT-8	the	The	a
DT-9	no	any	some
DT-10	an	a	the
DT-11	a	this	the

CD

CD-0	1	50	100
CD-1	8.50	15	1.2
CD-2	8	10	20
CD-3	1	30	31
CD-4	1989	1990	1988
CD-5	1988	1987	1990
CD-6	two	three	five
CD-7	one	One	Three
CD-8	12	34	14
CD-9	78	58	34
CD-10	one	two	three
CD-11	million	billion	trillion

PRP

PRP-0	It	He	I
PRP-1	it	he	they
PRP-2	it	them	him

RBR

RBR-0	further	lower	higher
RBR-1	more	less	More
RBR-2	earlier	Earlier	later

IN

IN-0	In	With	After
IN-1	In	For	At
IN-2	in	for	on
IN-3	of	for	on
IN-4	from	on	with
IN-5	at	for	by
IN-6	by	in	with
IN-7	for	with	on
IN-8	If	While	As
IN-9	because	if	while
IN-10	whether	if	That
IN-11	that	like	whether
IN-12	about	over	between
IN-13	as	de	Up
IN-14	than	ago	until
IN-15	out	up	down

RB

RB-0	recently	previously	still
RB-1	here	back	now
RB-2	very	highly	relatively
RB-3	so	too	as
RB-4	also	now	still
RB-5	however	Now	However
RB-6	much	far	enough
RB-7	even	well	then
RB-8	as	about	nearly
RB-9	only	just	almost
RB-10	ago	earlier	later
RB-11	rather	instead	because
RB-12	back	close	ahead
RB-13	up	down	off
RB-14	not	Not	maybe
RB-15	n’t	not	also

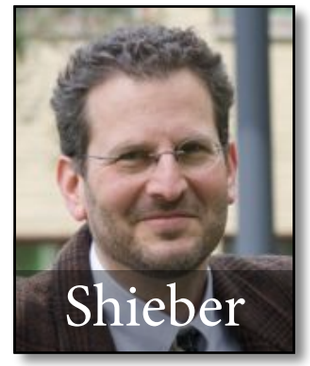
Summary

- PCFGs that we read off of treebank suffer from overly strong independence assumptions.

- Improve parser accuracy by encoding context in nonterminal vocabulary.
  - parent annotations
  - lexicalization
  - rule-based and automatically computed state splitting

- Berkeley parser: f-score around 90.

Parsing Schemata



- Parsing algorithm derives claims about the string. Record such claims in *parse items*.

- At each step, apply a *parsing rule* to infer new parse items from earlier ones.

- If there is a way to derive a *goal item* from the *start item(s)* for a given input string, then claim that this string is in the language.

Schema for shift-reduce

- Items are of the form (s,w') where w' is a suffix of the input string w, and s is the stack.
  - Claim of this item: Underlying cfg allows the derivation $s\ w' \Rightarrow^* w$

- Start item: (ε, w); goal item: (S, ε)

- Parsing rules:

$$\frac{(s,\ a \cdot w')}{(s \cdot a,\ w')}\ \text{(shift)} \qquad \frac{(s \cdot s',\ w') \quad A \to s' \text{ in } P}{(s \cdot A,\ w')}\ \text{(reduce)}$$

Implementing schemas

- Can generally implement parser for given schema in the following way:
  - maintain an *agenda:* queue of items that we have discovered, but not yet attempted to combine with other items
  - maintain a *chart* of all seen items for the sentence

```
initialize chart and agenda with all start items

while agenda not empty:
  item = dequeue(agenda)
  for each combination c of item with other item in the chart:
    if c not in chart:
      add c to chart
      enqueue c in agenda

if chart contains a goal item, claim w ∈ L(G)
```

rules of parsing schema used here

Implementing schemas

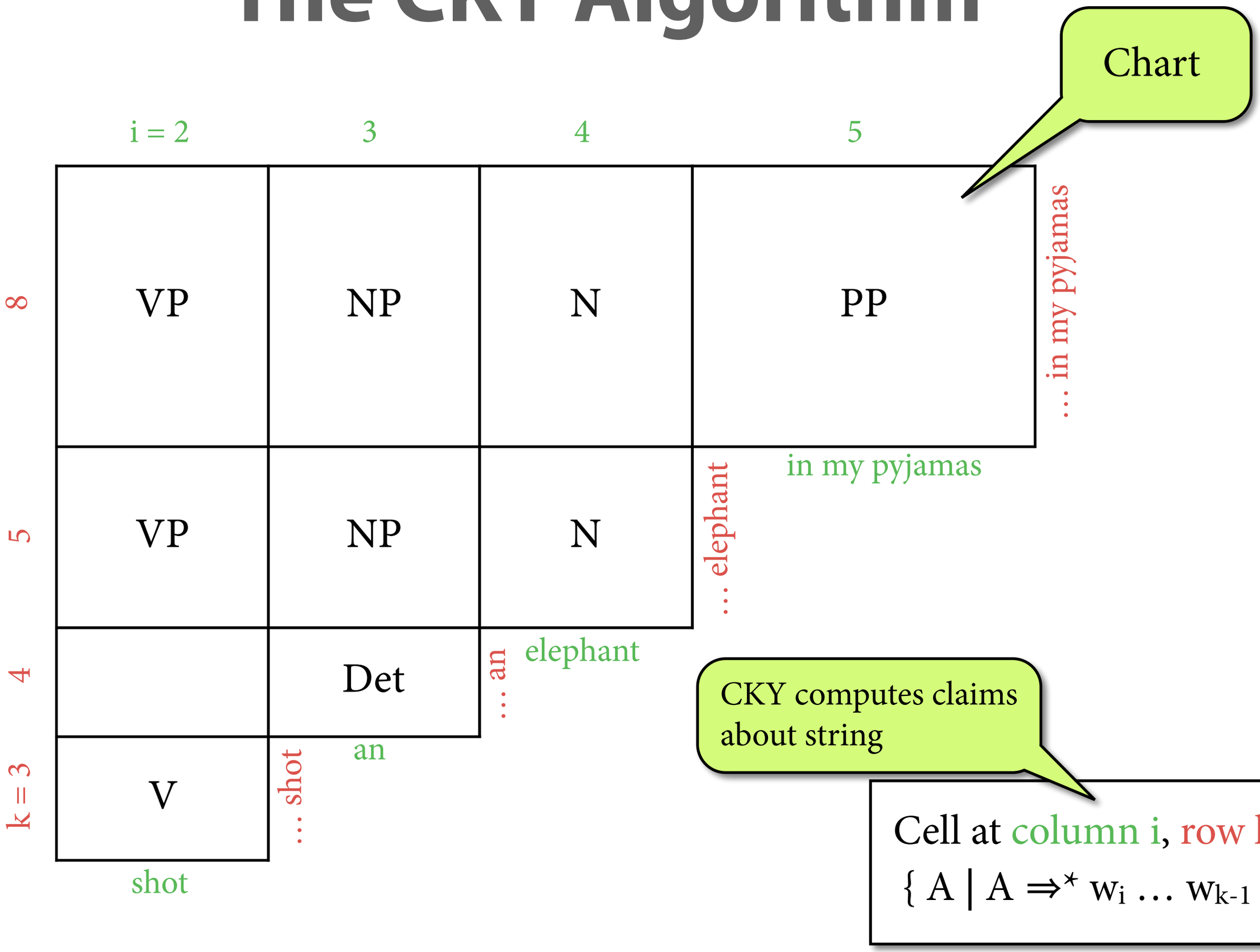
- Can generally implement parser for given schema in the following way:
  - maintain an *agenda:* queue of items that we have discovered, but not yet attempted to combine with other items
  - maintain a *chart* of all seen items for the sentence

```
initialize chart and agenda with all start items

while agenda not empty:
  item = dequeue(agenda)
  for each combination c of item with other item in the chart:
    if c not in chart:
      add c to chart
      enqueue c in agenda

if chart contains a goal item, claim w ∈ L(G)
```

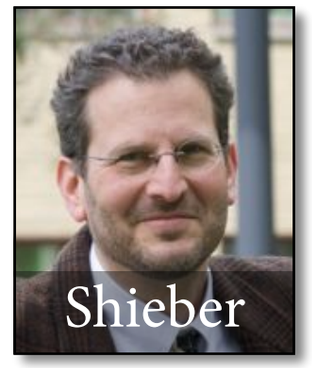
rules of parsing schema used here

essential to do this efficiently

The CKY Algorithm



CKY as parsing schema



- Makes *claims* about the string: Entering A into Ch(i,k) means algorithm thinks $A \Rightarrow^* w_i \ldots w_{k-1}$.

- Write this claim as *item* (A, i, k). This is like a logic formula that is true iff $A \Rightarrow^* w_i \ldots w_{k-1}$.

- Write *parsing schema* that shows how new items can be derived from old items.
  - very general view; applies to algorithms beyond CKY
  - supports generalized implementations

CKY as parsing schema

- Parsing schema for CKY has a single rule:

$$\frac{A \rightarrow B\ C \qquad (B, i, j) \qquad (C, j, k)}{(A, i, k)}$$

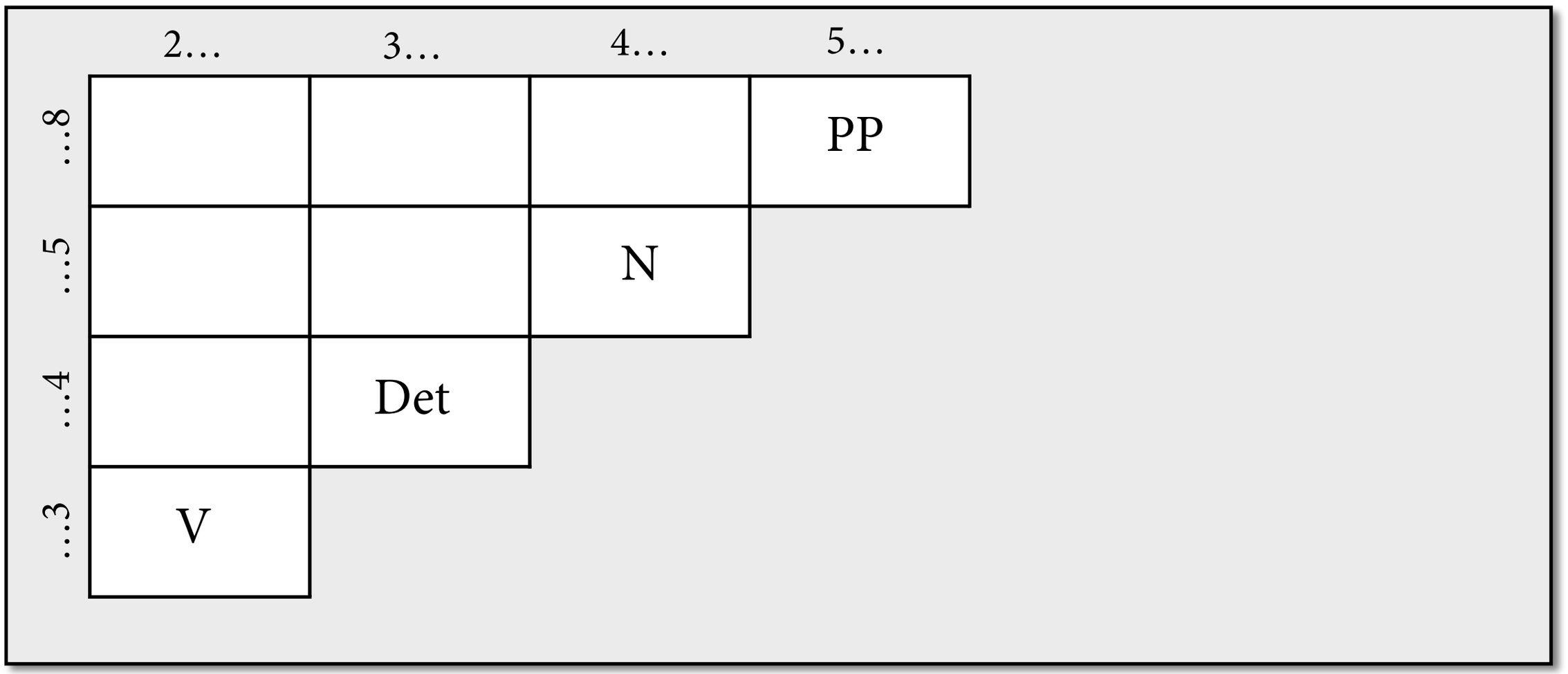
- One benefit: can literally read off parsing complexity.
  - rules have at most three independent variables for string positions (i, j, k)
  - therefore complexity is $O(n^3)$

Example

agenda:

(PP, 5, 8) (V, 2, 3) (Det, 3, 4) (N, 4, 5)

chart:

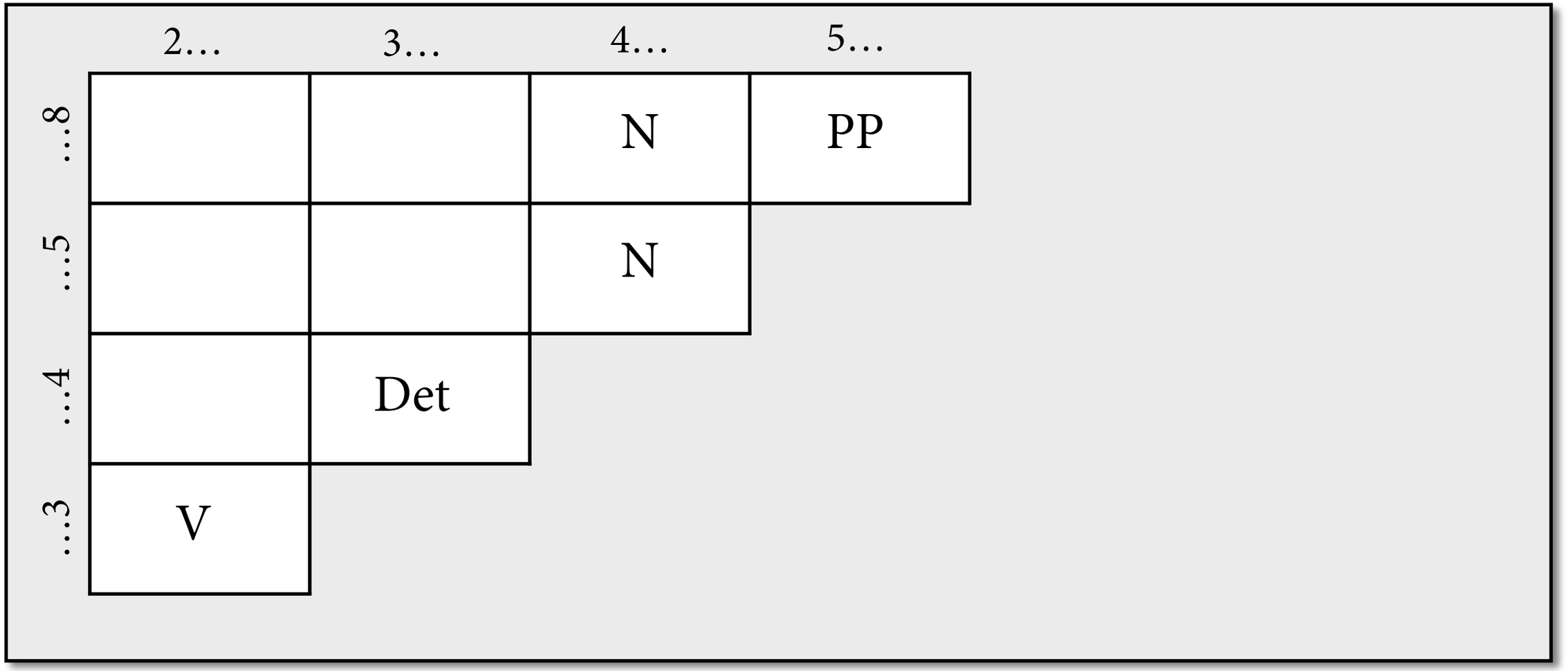


Example

agenda:

(PP, 5, 8) (V, 2, 3) (Det, 3, 4) (N, 4, 5) (N, 4, 8)

chart:

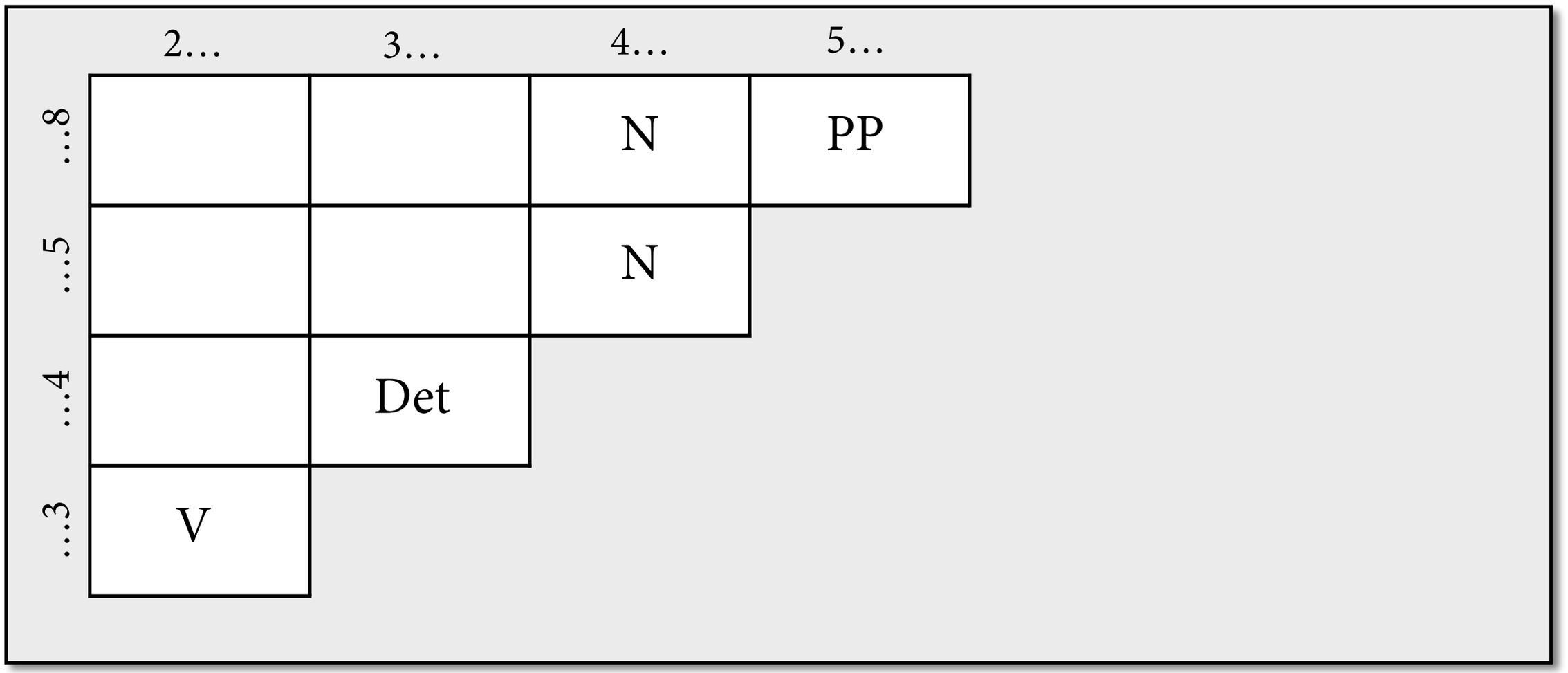


Example

agenda:

(V, 2, 3) (Det, 3, 4) (N, 4, 5) (N, 4, 8)

chart:

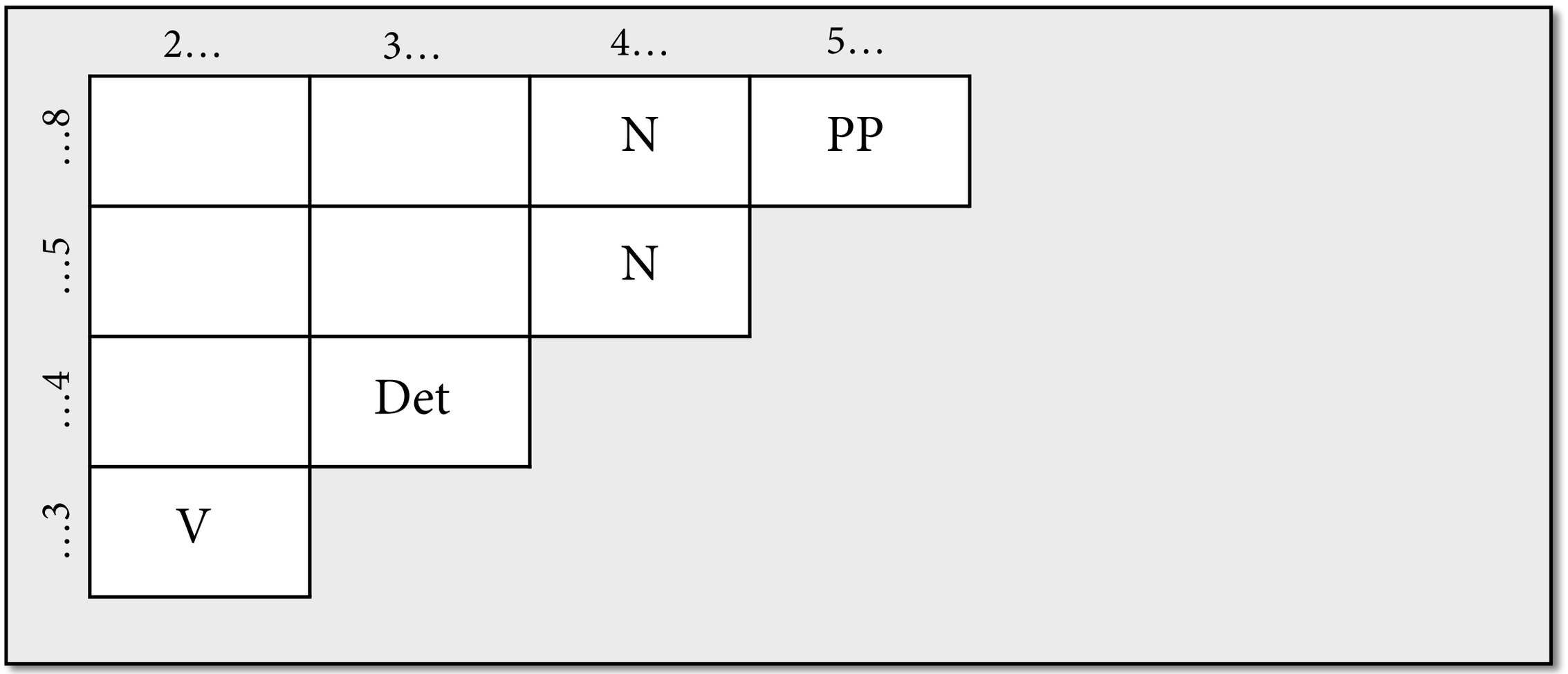


Example

agenda:

(Det, 3, 4) (N, 4, 5) (N, 4, 8)

chart:

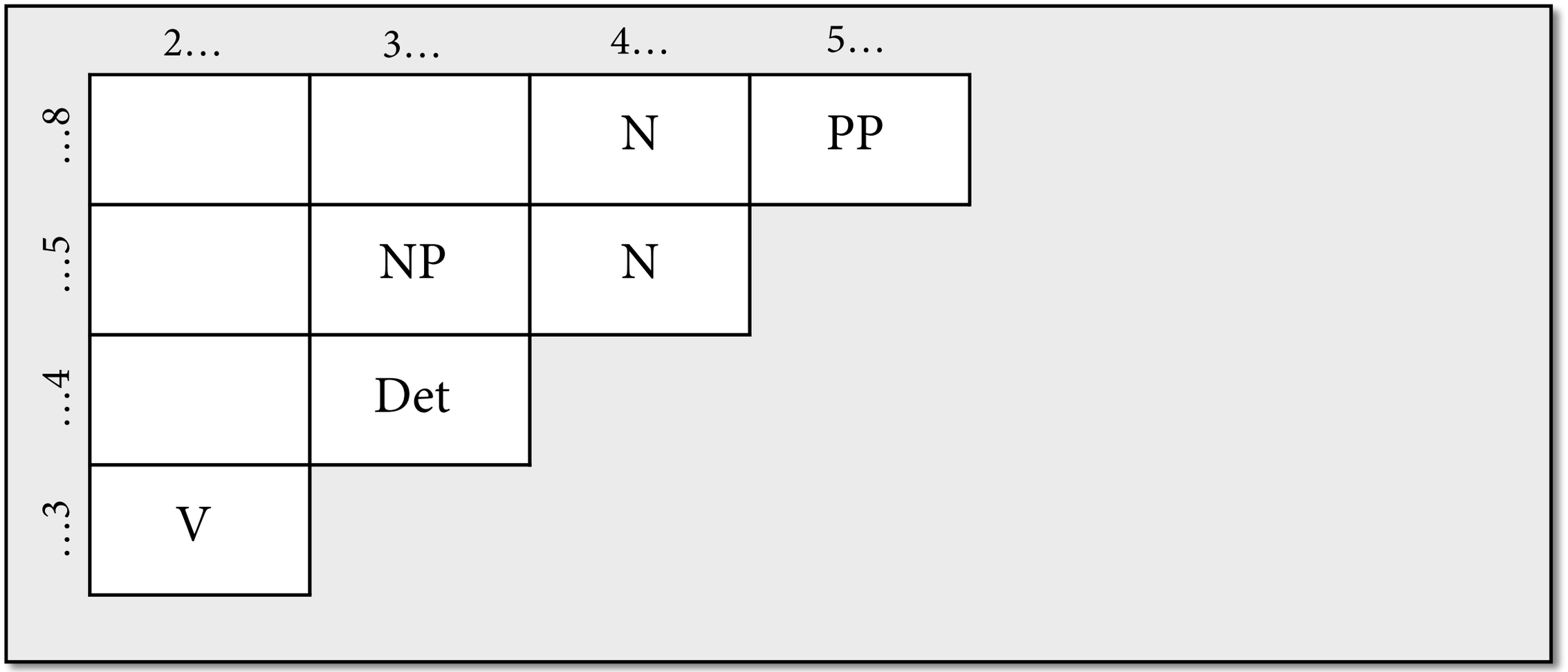


Example

agenda:

(Det, 3, 4) (N, 4, 5) (N, 4, 8) (NP, 3, 5)

chart:



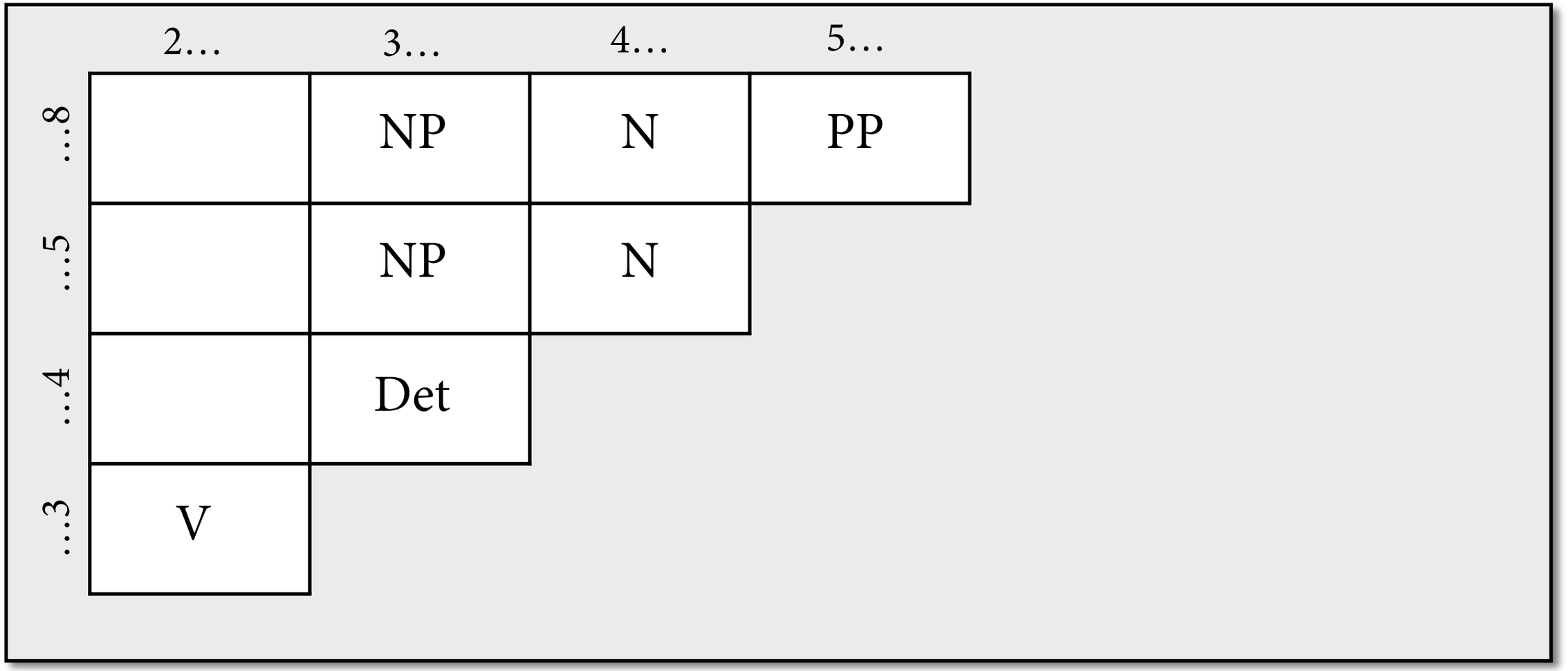
Example

agenda:

(Det, 3, 4) (N, 4, 5) (N, 4, 8) (NP, 3, 5)

(NP, 3, 8)

chart:



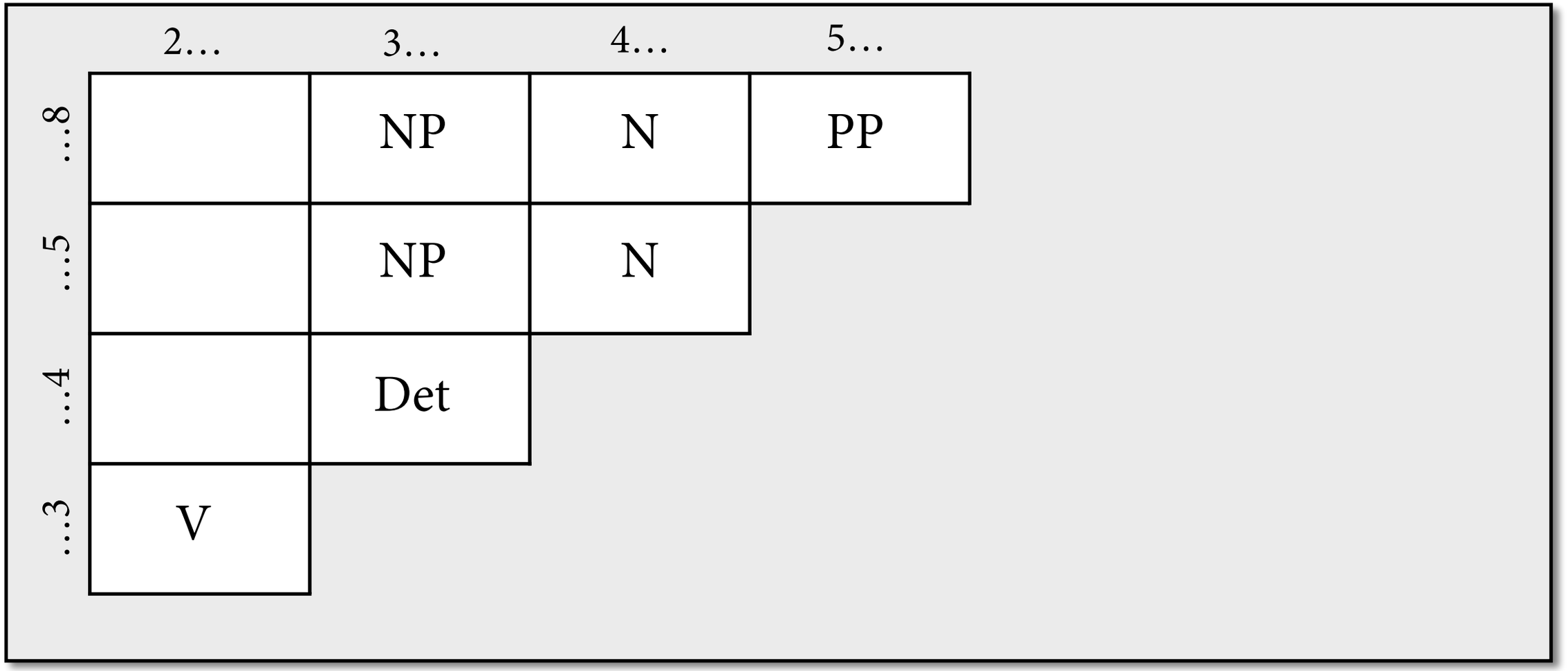
Example

agenda:

(N, 4, 5) (N, 4, 8) (NP, 3, 5)

(NP, 3, 8)

chart:



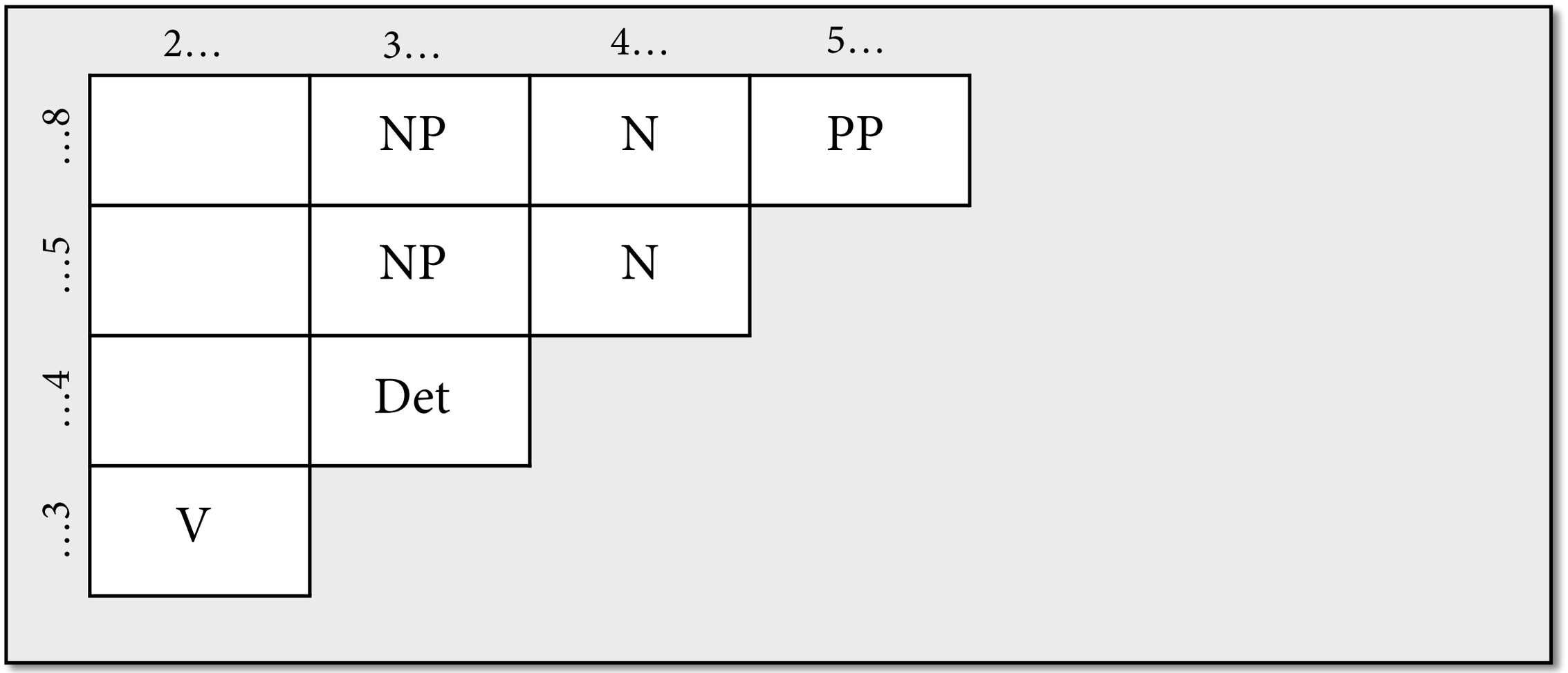
Example

agenda:

(N, 4, 8) (NP, 3, 5)

(NP, 3, 8)

chart:



Example

agenda:

(NP, 3, 5)

(NP, 3, 8)

chart:

	2…	3…	4…	5…
…8		NP	N	PP
…5		NP	N	
…4		Det		
…3	V			

Example

agenda:

(NP, 3, 5)

(NP, 3, 8) (VP, 2, 5)

chart:

	2…	3…	4…	5…
…8		NP	N	PP
…5	VP	NP	N	
…4		Det		
…3	V			

Example

agenda:

(NP, 3, 8) (VP, 2, 5)

chart:

	2…	3…	4…	5…
…8		NP	N	PP
…5	VP	NP	N	
…4		Det		
…3	V			

Example

agenda:

(NP, 3, 8) (VP, 2, 5) (VP, 2, 8)

chart:

	2…	3…	4…	5…
…8	VP	NP	N	PP
…5	VP	NP	N	
…4		Det		
…3	V			

Example

agenda:

(VP, 2, 5) (VP, 2, 8)

chart:

	2…	3…	4…	5…
…8	VP	NP	N	PP
…5	VP	NP	N	
…4		Det		
…3	V			

Example

agenda:

(VP, 2, 8)

chart:

	2…	3…	4…	5…
…8	VP	NP	N	PP
…5	VP	NP	N	
…4		Det		
…3	V			

Example

agenda:

chart:

	2…	3…	4…	5…
…8	VP	NP	N	PP
…5	VP	NP	N	
…4		Det		
…3	V			