# Computational Linguistics

CSC 2501 / 485
Fall 2015

3

## 3. Chart parsing

Frank Rudzicz
Toronto Rehabilitation Institute-UHN; and
Department of Computer Science, University of Toronto

Reading: Jurafsky & Martin: 13.3–4.
Allen: *3.4, 3.6.*
Bird et al.: 8.4, online extras 8.2 to end of section "Chart Parsing in NLTK".

# Efficient parsing

- Want to avoid problems of blind search:
  - Avoid redoing analyses that are identical in more than one path of the search.
- Guide the analysis with both
  - the actual input
  - the expectations that follow from the choice of a grammar rule.
- Combine strengths of both top-down and bottom-up methods.

# Chart parsing

- Main idea:
  - Use data structures to maintain information:
    a **chart** and an **agenda**
- **'Agenda':**
  - List of constituents that need to be processed.
- **'Chart':**
  - Records ("***memorizes***") work; obviates repetition.
  - Related to: Well-formed substring table (WFST);
    CKY parsing;
    Earley parsing;
    dynamic programming.

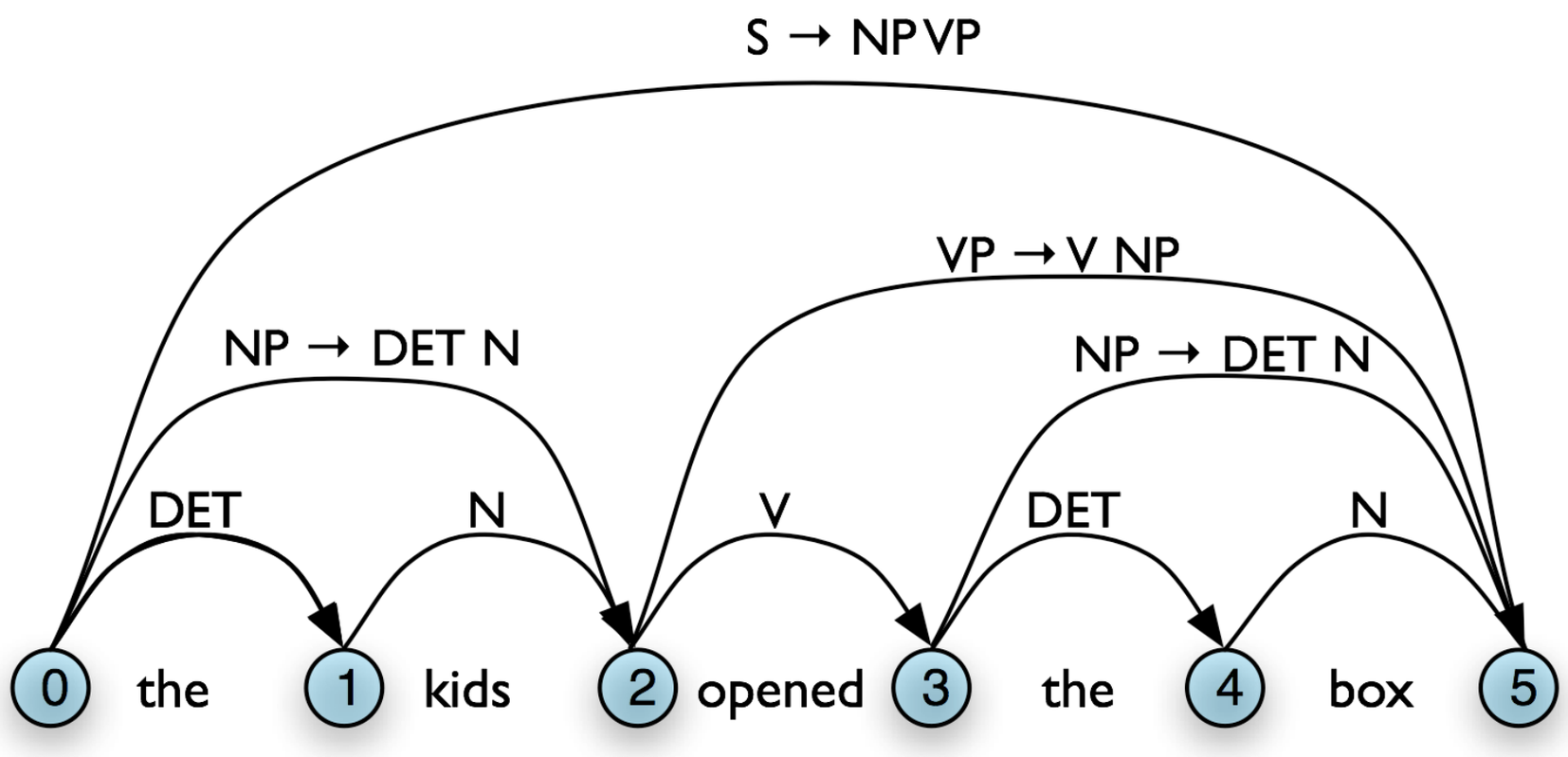
# Charts 1

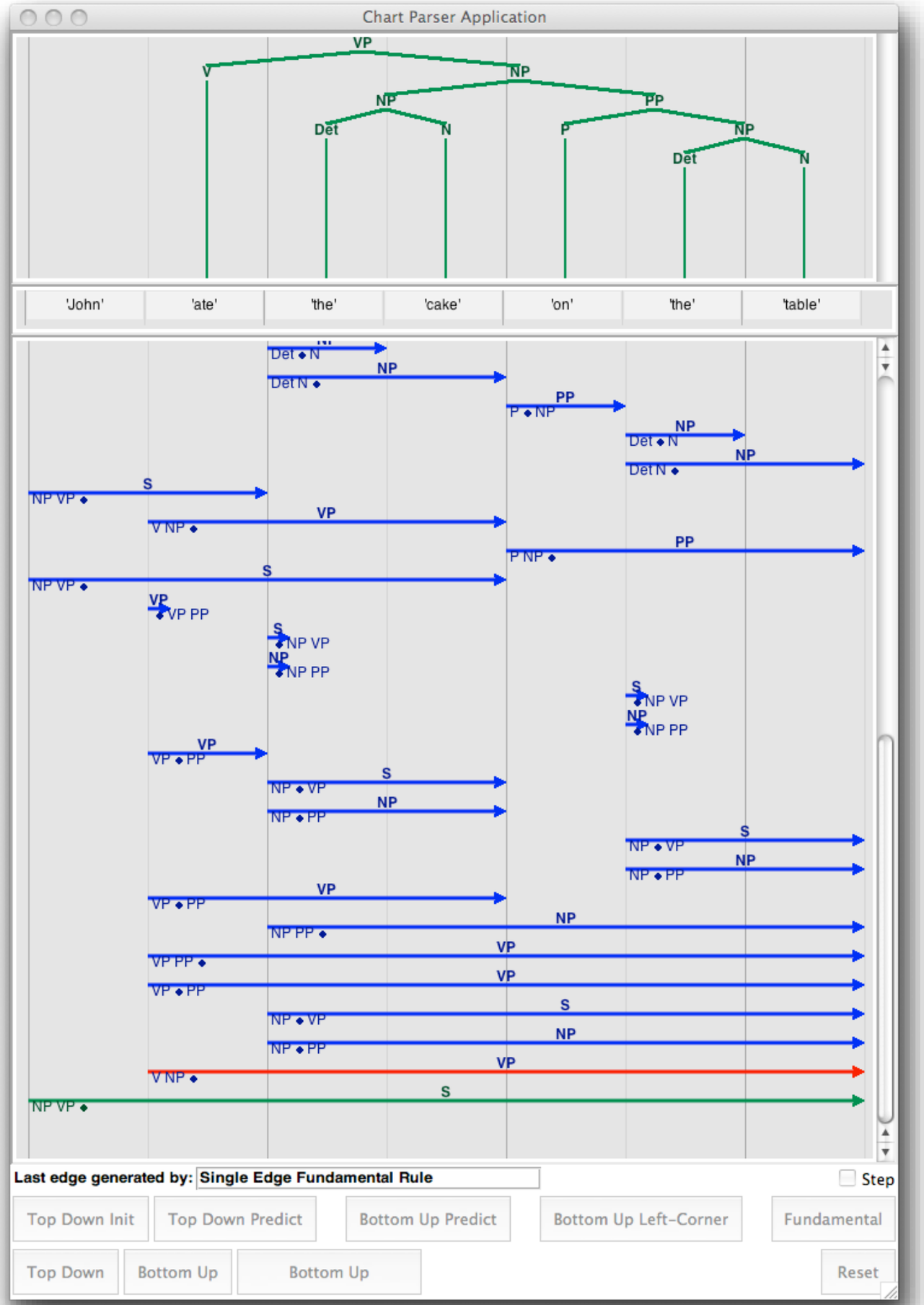
- Contents of chart:
  1. Partially built constituents (also called **active arcs**).
     Think of them as ***hypotheses***.
  2. Completed constituents (**inactive arcs**).

- Representation: Labelled arc (**edge**) from one point in sentence to another (or same point).
  - Directed; always left-to-right (or to self).
  - Label is grammar rule used for arc.

# Charts 2

- Notation for positions in sentence from 0 to *n* (length of sentence):
- $_0$ *The* $_1$ *kids* $_2$ *opened* $_3$ *the* $_4$ *box* $_5$

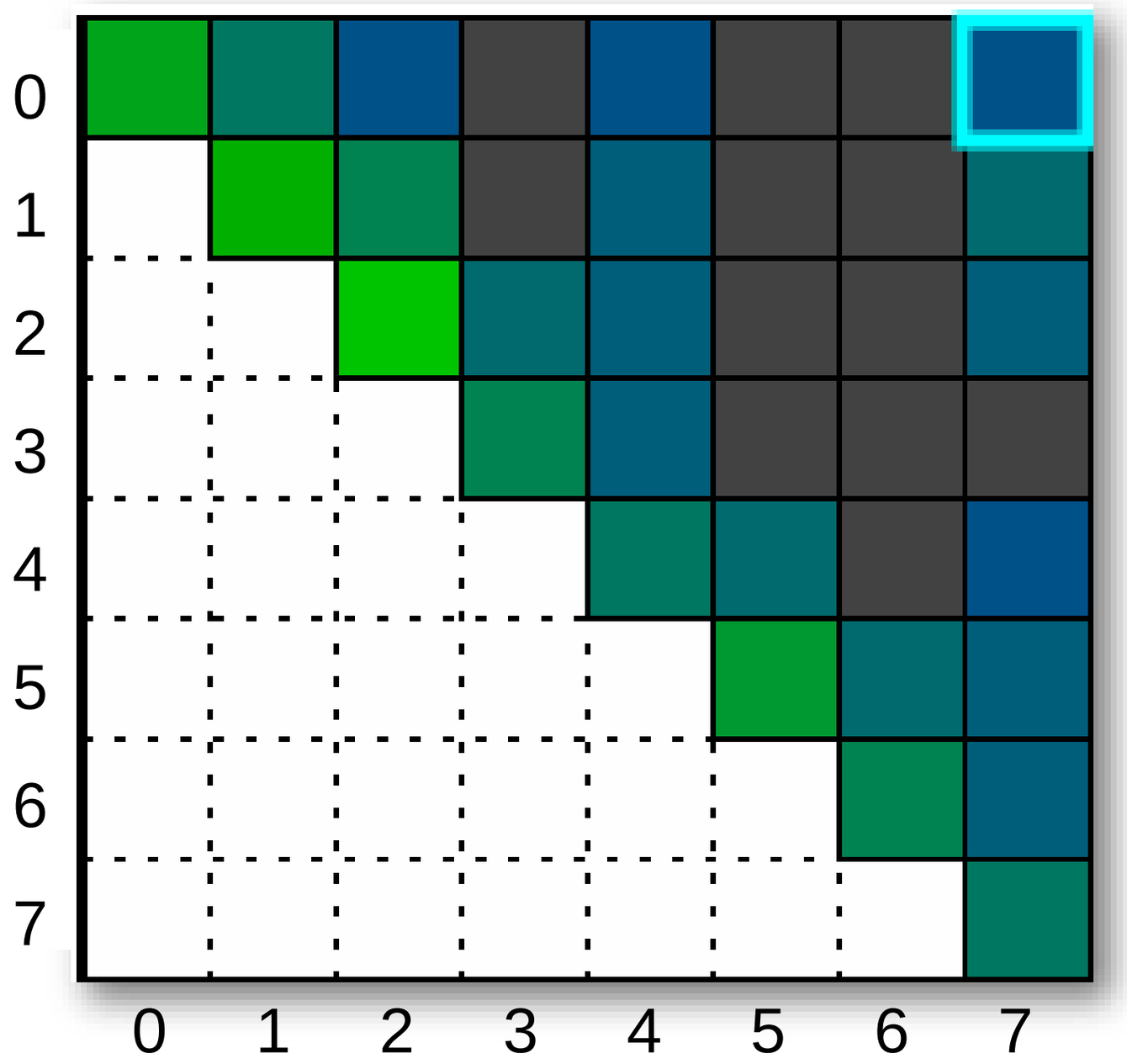





Part of a chart from the NLTK chart parser demo, `nltk.app.chartparser()`

# Charts 3

- An arc can connect any positions $i, j$ $(0 \leq i \leq j \leq n)$.
- You can have $> 1$ arc on any $i, j$.
- You can associate all arcs on positions $i, j$ with cell $ij$ of upper-triangular matrix.



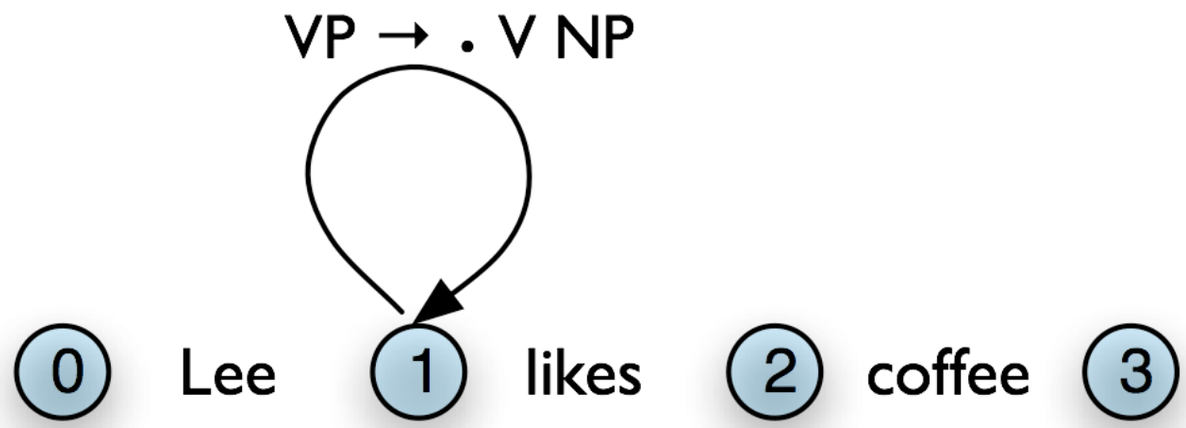

Arcs in top right corner cell cover the whole sentence.

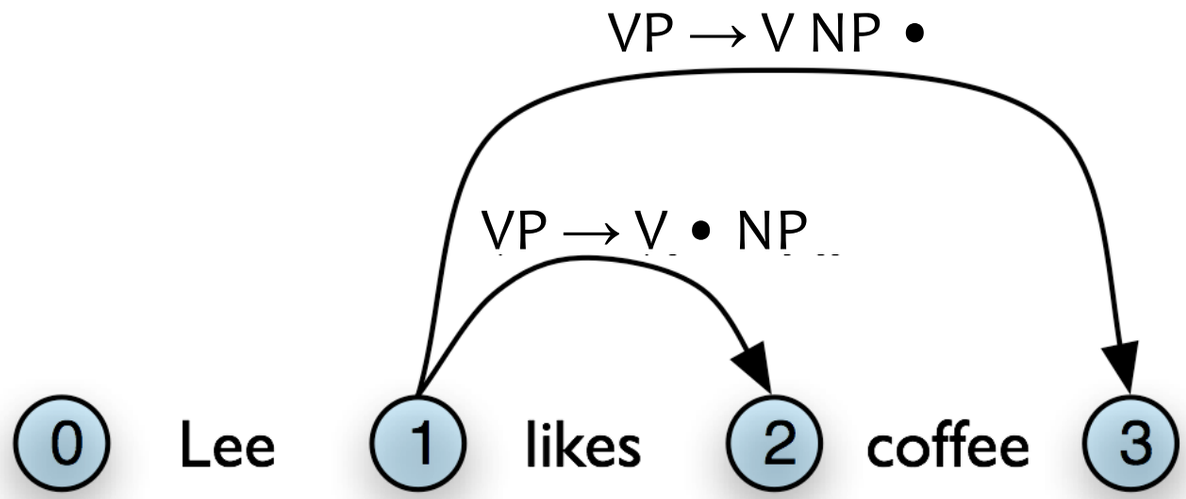
Those for S are **'parse edges'**.

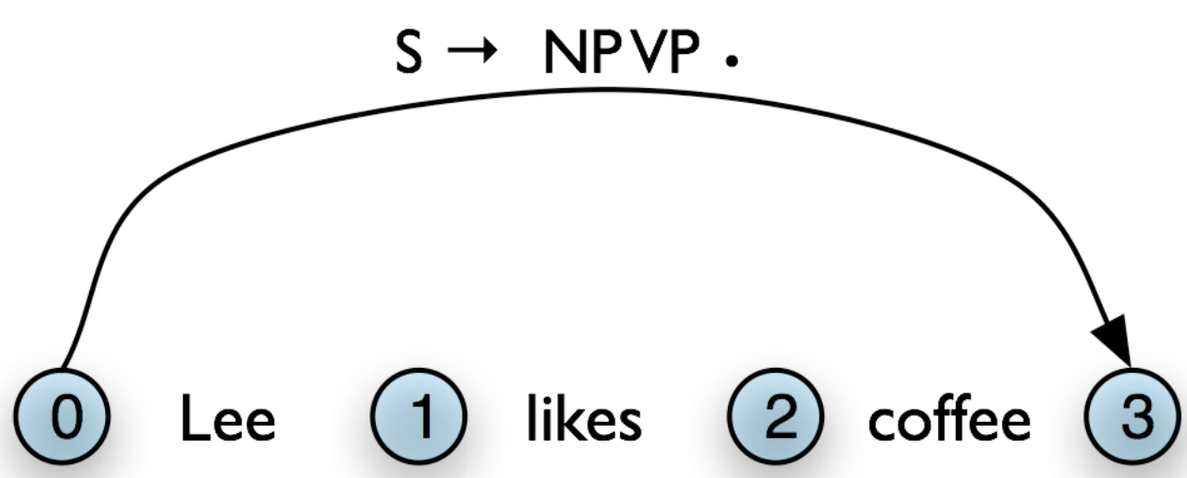
The matrix for a seven-word sentence
from the NLTK chart parser demo
`nltk.app.chartparser()`

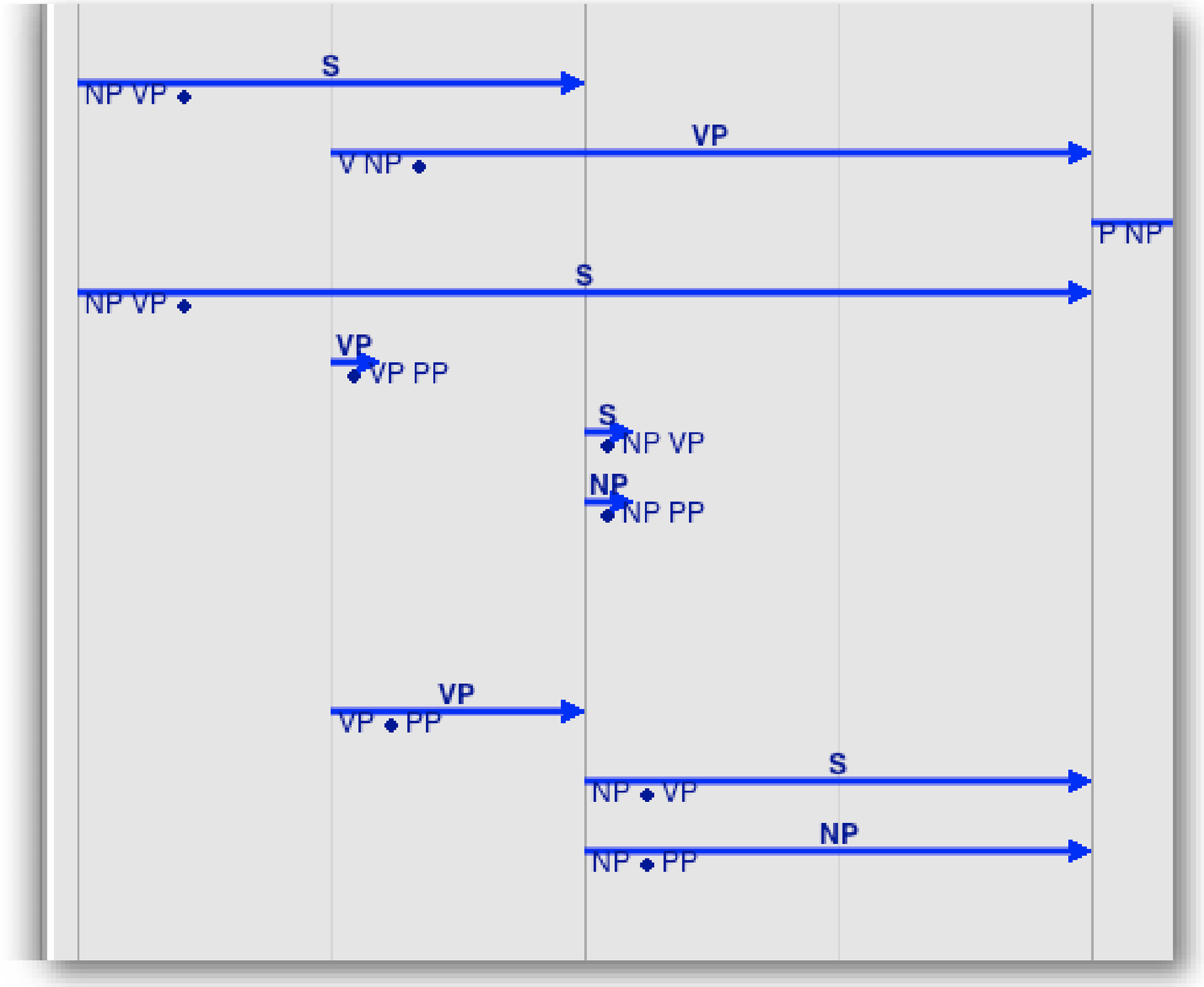
# Notation for arc labels

- *Notation*: '•' means 'complete to here'.
  - A → X Y • Z **(*active*)**
    *'In parsing an A, we've so far seen an X and a Y, and our A will be complete once we've seen a Z.'*
  - A → X Y Z • **(*inactive*)**
    *'We have seen an X, a Y, and a Z, and hence completed the parse of an A.'*
  - A → • X Y Z **(*active*)**
    *'In parsing an A, so far we haven't seen anything.'*






Part of a chart from the NLTK chart parser demo,
`nltk.app.chartparser()`

# Fundamental rule of chart parsing

- **Arc extension:**

Let *X*, *Y*, *and* *Z* be sequences of symbols, where *X* and *Y* are possibly empty.

**If** the chart contains an **active** arc from ***i*** to *j* of the form

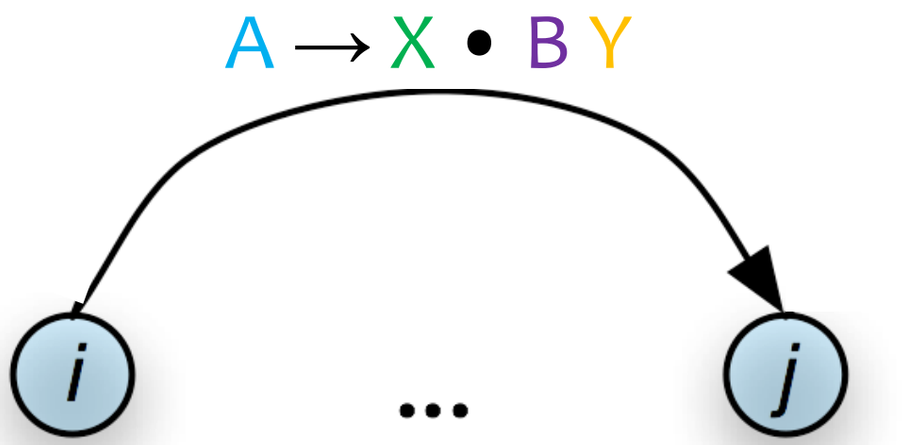
A → X • B Y

**and** a completed arc from *j* to ***k*** of the form

B → Z • or B → *word*

**then** add an arc from *i* to *k*

A → X B • Y



# Bottom-up arc-addition rule

- **Arc addition** (or **prediction**):

If the chart contains an completed arc from *i* to *j* of the form

$$A \rightarrow X \bullet$$

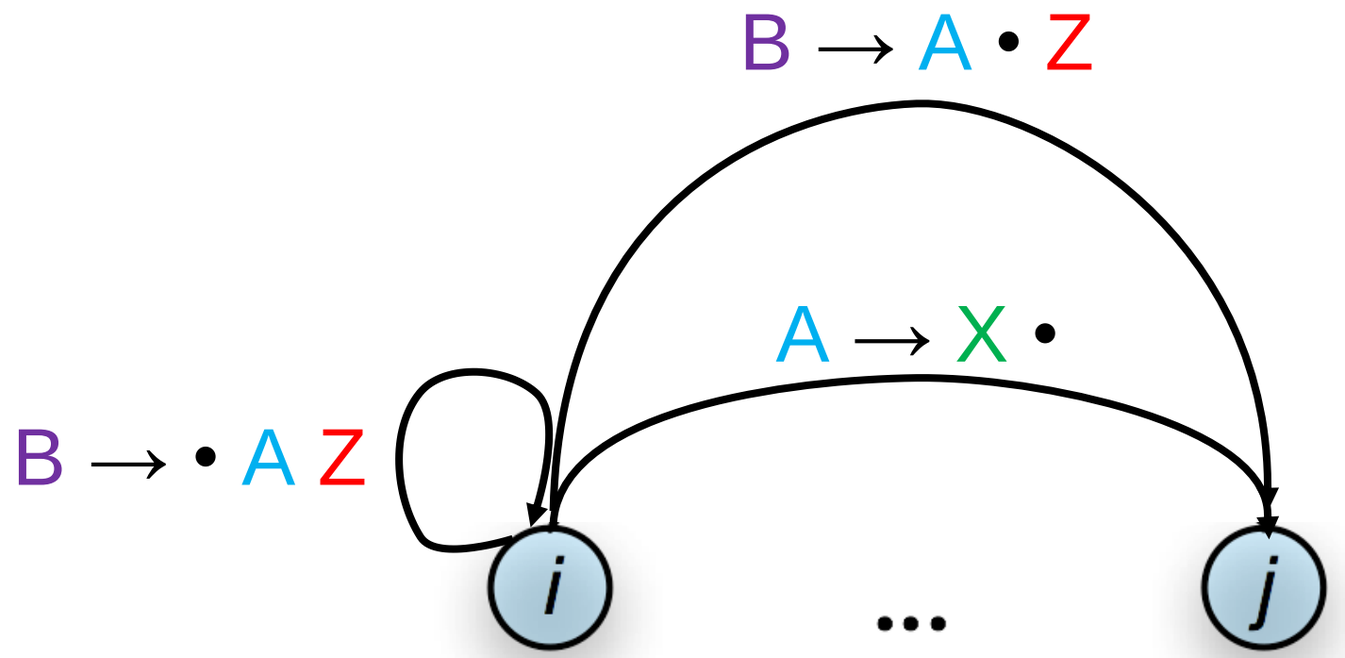
and the grammar contains a rule

$$B \rightarrow A\ Z$$

then add an arc from *i* to *i* (***reflexive***)

$$B \rightarrow \bullet\ A\ Z$$

or an arc $B \rightarrow A \bullet Z$ from *i* to *j*.

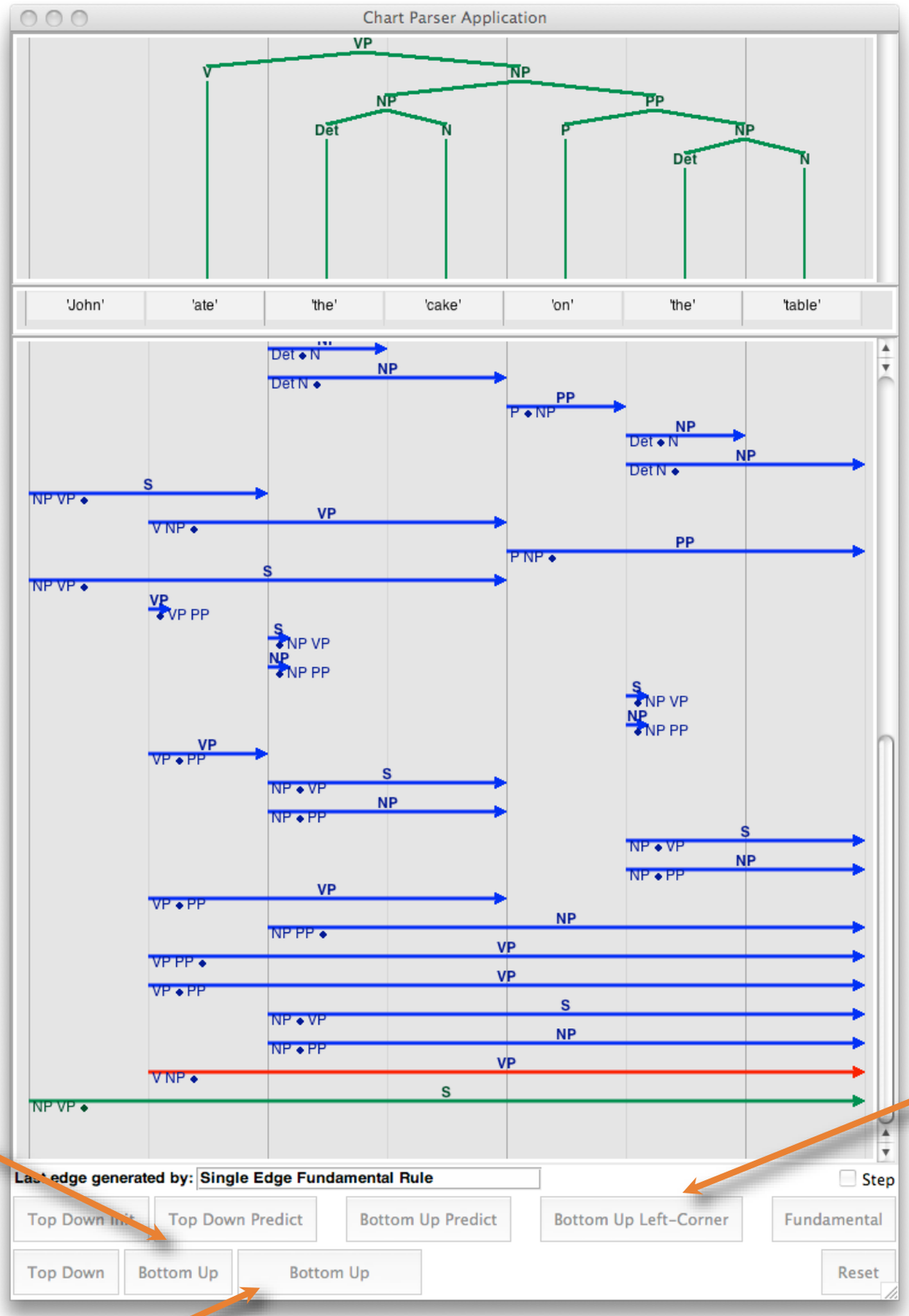



Adapted from: Steven Bird, Ewan Klein, and Edward Loper, *Natural Language Processing in Python*, v. 9.5.3, July 2008. Used under Creative Commons licence.

# Bottom-up chart parsing
## BKL's view

- Initialize chart with **each word** in the **input** sentence.
- Until nothing more happens:
  - Apply the **bottom-up addition** rule wherever you can.
  - Apply the **fundamental rule** wherever you can.
- Return the trees corresponding to the parse edges in the chart.

# >>> nltk.app.chartparser()




Top-down Init Rule

Top-down Predict Rule

Top-down Strategy

Bottom-up Strategy

Bottom-up Left-Corner Strategy

Bottom-up Predict Rule

Bottom-up Left-Corner Predict Rule

Fundamental Rule

Reset Parser

# Observations

- This cool thing builds all constituents exactly once.
- It never re-computes the prefix of an RHS.
- It exploits context-free nature of rules to reduce the search. How?

# Controlling the process

- Doing everything you can is too uncontrolled.
- Try to avoid predictions and expansions that will lead nowhere, dummy.
- So use an **agenda** — a list of completed arcs.
  - When an arc is completed, it is initially added to the agenda, not the chart.
  - **Agenda rules** *decide* which completed arc to move to the chart next.
  - *E.g.*, treat agenda as *stack* or as *queue*; or pick item that looks "most efficient" or "most likely"; or pick NPs first; or ….

# Bottom-up chart parsing
# Jurafsky & Martin's view

- Initialize agenda with the list of **lexical categories** (Pos) of each word in the input sentence.
- Until **agenda** is **empty**, repeat:

—Move next constituent C from agenda to chart.

i. Find rules whose RHS starts with C and add corresponding active arcs to the chart.

ii. Find active arcs that continue with C and extend them; add the new active arcs to the chart.

iii. Find active arcs that have been completed; add their LHS as a new constituent to the agenda.

# Bottom-up chart parsing
# Algorithm the first

INITIALIZE:
**set** *Agenda* = list of all possible categories of each input word
(in order of input);
**set** *n* = length of input;
**set** *Chart* = ();

ITERATE:
**loop**
**if** *Agenda* = () **then**
**if** there is at least one S constituent from 0 to *n* **then**
**return** SUCCESS ☺
**else**
**return** FAIL ☹
**end if**
**else …**

# Bottom-up chart parsing
# Algorithm the second

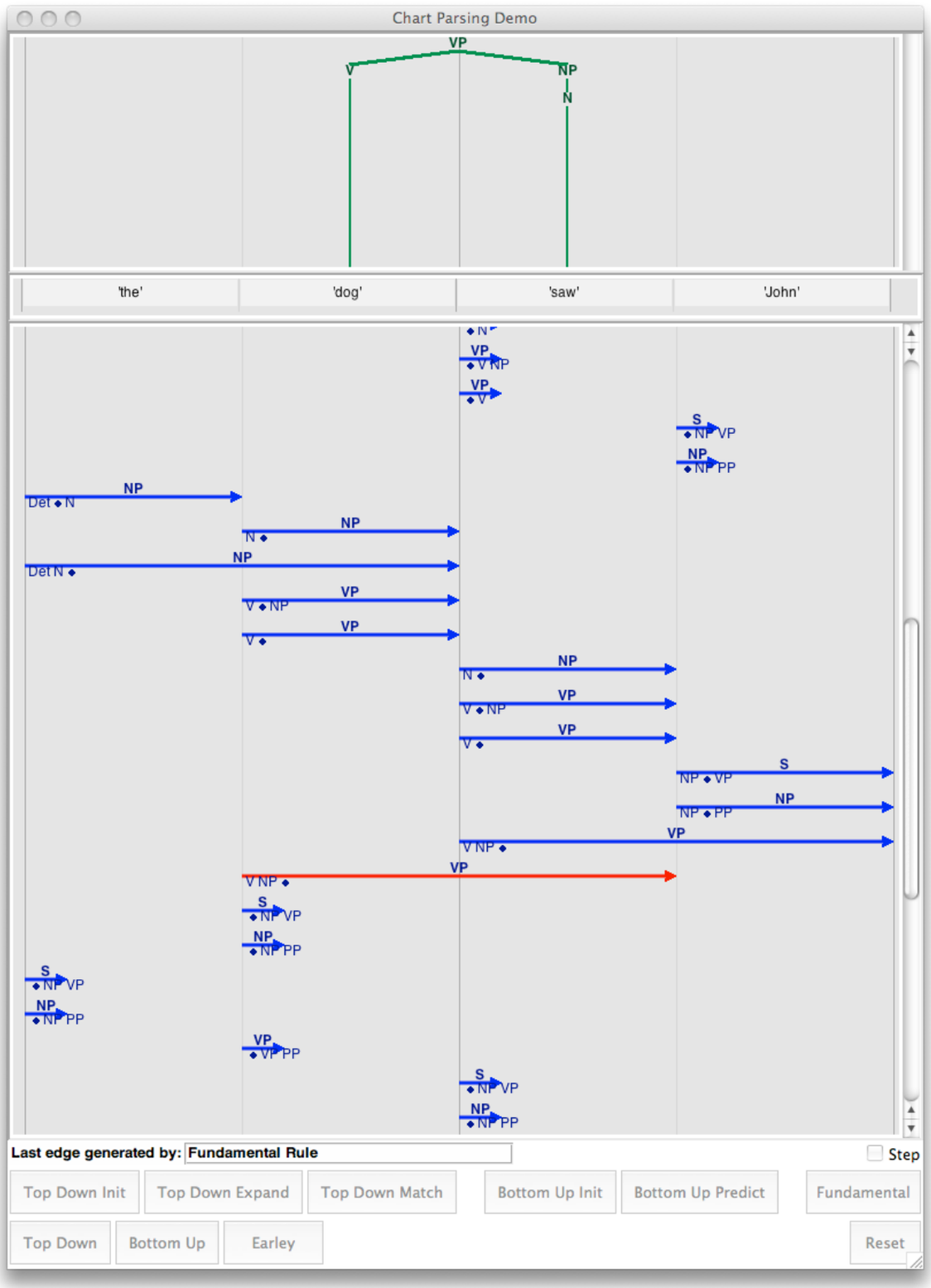
**Set** $C_{i,j}$ = First(*Agenda*); /* *Remove first item from agenda.* */
/* $C_{i,j}$ *is a completed constituent of type C from position i to position j* */
Add $C_{i,j}$ to *Chart*;

ARC UPDATE:
a. BOTTOM-UP ARC ADDITION (PREDICTION):
**for each** grammar rule X → C X1 … XN **do**
Add arc X → C • X1 … XN, from *i* to *j*, to *Chart;*
b. ARC EXTENSION (FUNDAMENTAL RULE):
**for each** arc X → X1 … • C … XN, from *k* to *i*, **do**
Add arc X → X1 … C • … XN, from *k* to *j*, to *Chart;*
c. ARC COMPLETION:
**for each** arc X → X1 … XN C • added in step (a) or step (b) **do**
Move completed constituent X to *Agenda*;
**end if**
**end loop**

# Problem with bottom-up chart parsing

- It ignores useful top-down knowledge (rule contexts).

# >>> nltk.app.chartparser()




Add lexical ambiguity to defaults:

N → *saw*

V → *dog*

NP → N

Parse bottom-up:

*the dog saw John*

# Top-down chart parsing

- Same as bottom-up, except **new** arcs are added to chart ***only*** if they are based on predictions from **existing** arcs.

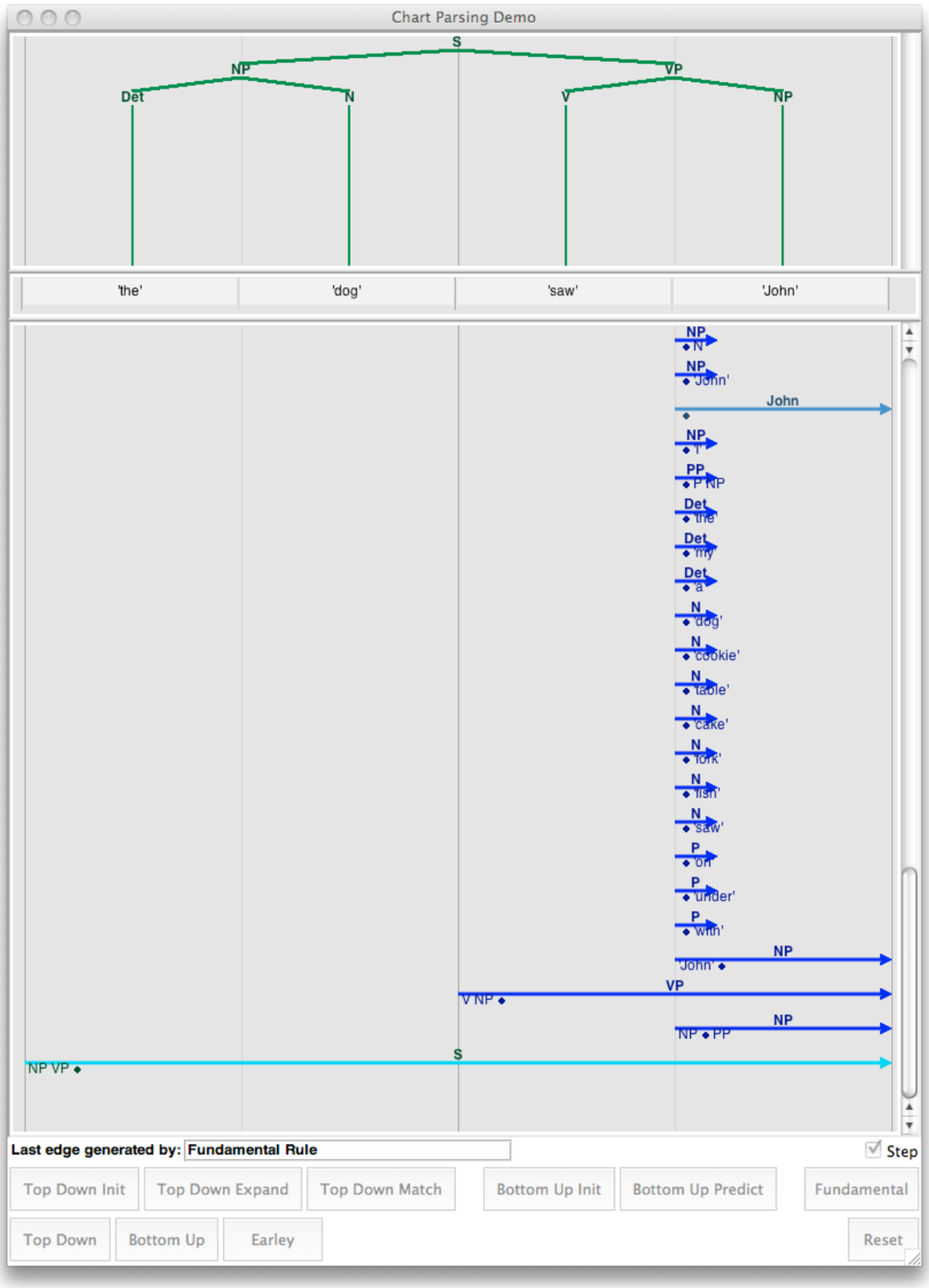
- Initialize chart with unstarted active arcs for S.

  S → • X Y

  S → • Z Q

- Whenever an active arc is added, also add unstarted arcs for its next needed constituent.

# >>> nltk.app.chartparser()




Add lexical ambiguity to defaults:

N → *saw*

V → *dog*

NP → N

Parse top-down:

*the dog saw John*

# Top-down chart parsing

## Algorithm the first

INITIALIZE:

**set** *Agenda* = list of all possible categories of each input word
(in order of input);
**set** *n* = length of input;
**set** *Chart* = ();
**for each** grammar rule S → X1 … XN **do**
Add arc S → • X1 … XN to *Chart* at position 0;
apply TOP-DOWN ARC ADDITION [step (a') below] to the new arc;
**end for**

ITERATE:

**loop**
**if** *Agenda* = () **then**
**if** there is at least one *S* constituent from 0 to *n* **then**
**return** SUCCESS
**else**
**return** FAIL
**end if**
**else …**

# Top-down chart parsing
# Algorithm the second

**Set** $C_{i,j}$ = First(*Agenda*);  /* *Remove first item from agenda.* */
/* $C_{i,j}$ *is a completed constituent of type C from position i to position j* */
Add $C_{i,j}$ to *Chart*;

ARC UPDATE:
b. ARC EXTENSION (FUNDAMENTAL RULE):
**for each** arc X → X1 … • C … XN, from *k* to *i*, **do**
Add arc X → X1 … C • … XN, from *k* to *j*, to *Chart;*
a'. TOP-DOWN ARC ADDITION (PREDICTION):
/* *Recursive: until no new arcs can be added* */
**for each** arc X → X1 … • XL … XN, from *k* to *j*, added in step (b) or (a'), **do**
Add arc XL → • Y1 … YM, from *j* to *j*, to *Chart*;
c. ARC COMPLETION:
**for each** arc X → X1 … XN C • added in step (b) **do**
Move completed constituent X to *Agenda*;
**end if**
**end loop**

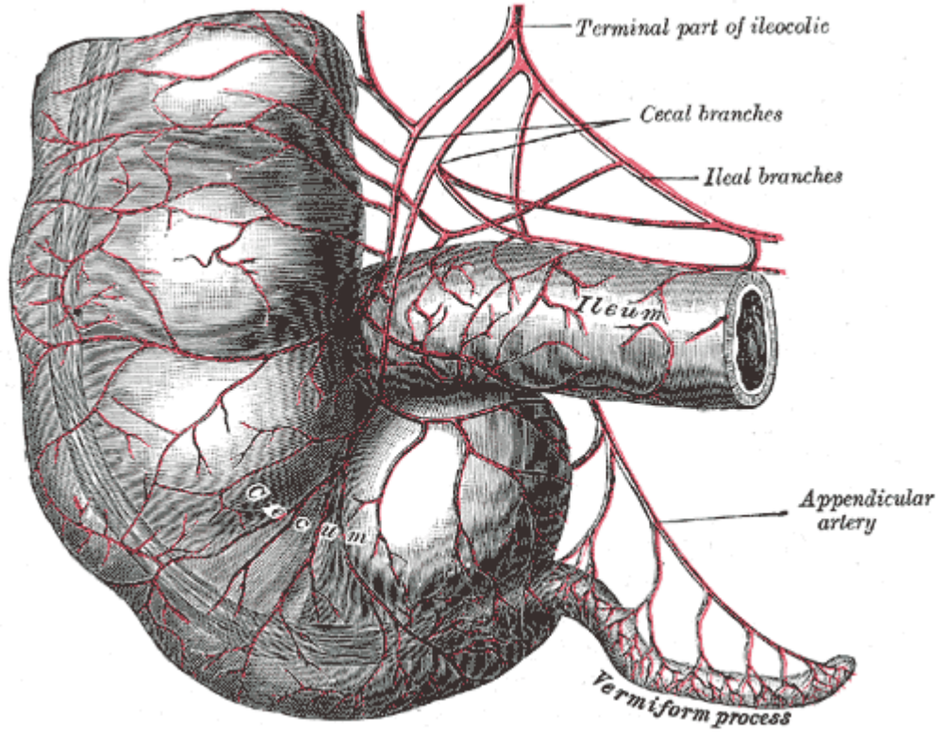
# Notes on chart parsing

- Chart parsing separates:
  1. Policy for selecting constituent from agenda;
  2. Policy for adding new arcs to chart;
  3. Policy for initializing chart and agenda.

- "Top-down" and "bottom-up" now refer to arc-addition rule.
  - Initialization rule gives bottom-up aspect in either case.

- Polynomial algorithm ($\Theta(n^3)$), instead of exponential.

# Appendix


Terminal part of ileocolic
Cecal branches
Ileal branches
Ileum
Appendicular artery
Vermiform process

1 Paint 2 the 3 wall 4 with 5 cracks 6

1 2 3 4 5 6

Chart: Completed arcs

Chart: Active arcs

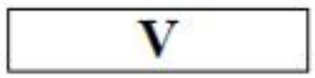
Grammar

S → V NP
S → V NP PP
NP → (Art) N
NP → NP PP
PP → P NP

{V, Art, N, P, N}

Agenda

1 Paint 2 the 3 wall 4 with 5 cracks 6




1 2 3 4 5 6

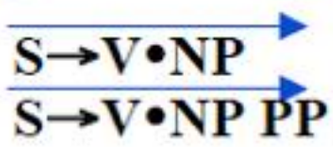



Chart: Completed arcs

Chart: Active arcs

Grammar

S → V NP
S → V NP PP
NP → (Art) N
NP → NP PP
PP → P NP

{V, Art, N, P, N}
{Art, N, P, N}

Agenda

1 Paint 2 the 3 wall 4 with 5 cracks 6

| V | Art |
|---|---|

1 2 3 4 5 6

S→V•NP NP→Art•N

S→V•NP PP

Chart: Completed arcs

Chart: Active arcs

Grammar

S → V NP
S → V NP PP
NP → (Art) N
NP → NP PP
PP → P NP

Agenda

{V, Art, N, P, N}
{Art, N, P, N}
{N, P, N}

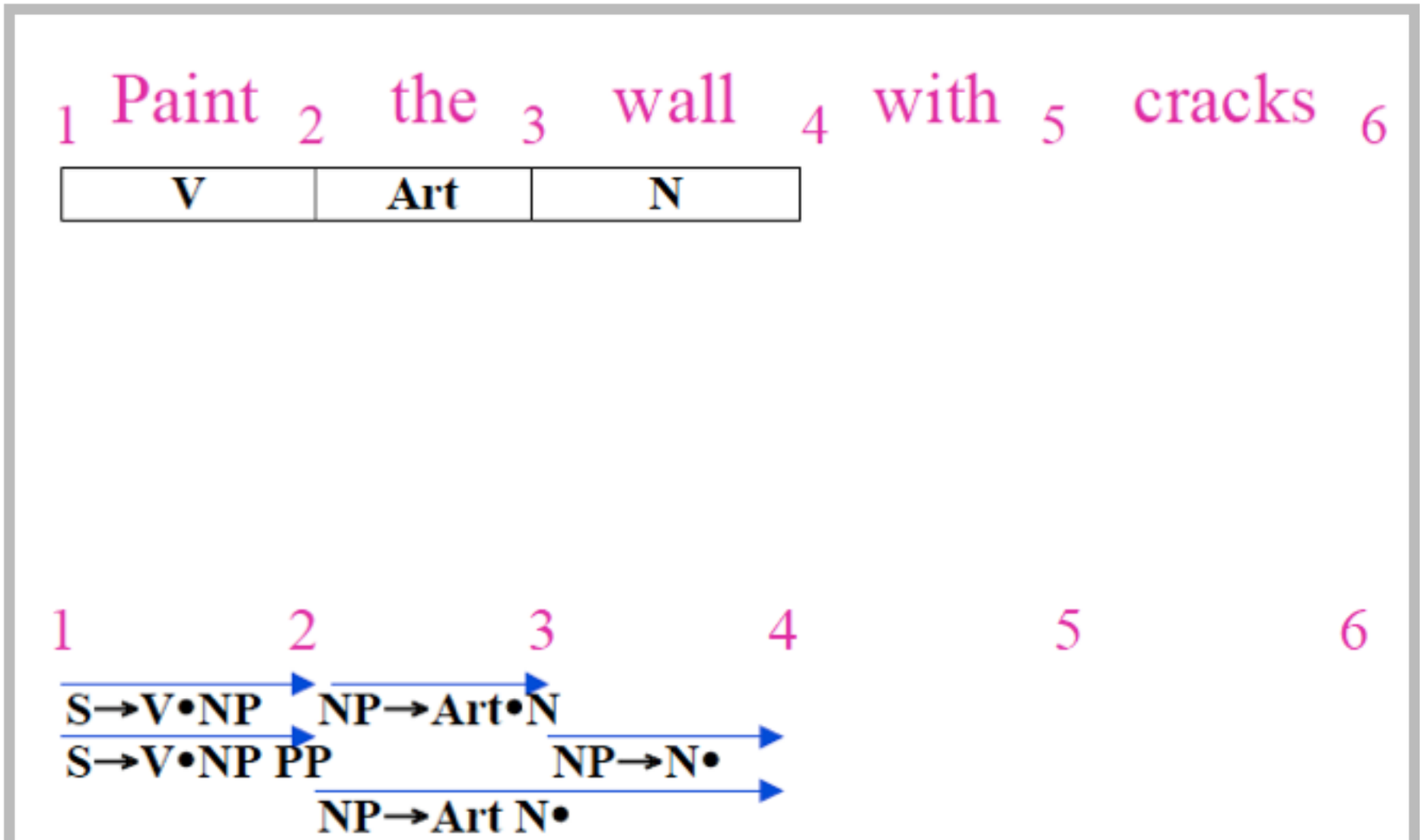



Chart: Completed arcs

Chart: Active arcs

Grammar

S → V NP
S → V NP PP
NP → (Art) N
NP → NP PP
PP → P NP

{V, Art, N, P, N}
{Art, N, P, N}
{N, P, N}
{P, N}
{NP, NP, P, N}

Agenda

1 Paint 2 the 3 wall 4 with 5 cracks 6

| V | Art | N |
|---|---|---|
| | $NP_1$ | |

1 2 3 4 5 6

S→V•NP

S→V•NP PP

NP→Art•N

NP→N•

NP→Art N•

S→V $NP_1$•

S→V $NP_1$•PP

NP→$NP_1$•PP

Chart: Completed arcs

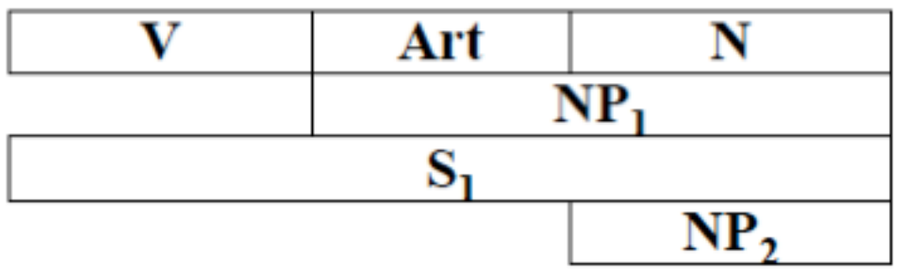
Chart: Active arcs

Grammar

S → V NP
S → V NP PP
NP → (Art) N
NP → NP PP
PP → P NP

{V, Art, N, P, N}
{Art, N, P, N}
{N, P, N}
{P, N}
{NP, NP, P, N}
{NP, P, N}
{S, NP, P, N}

Agenda



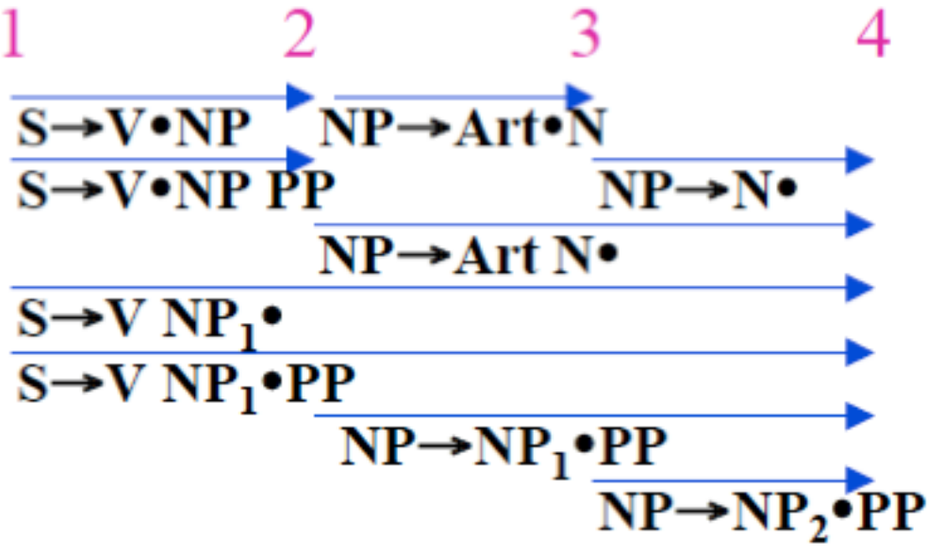




Chart: Completed arcs

Chart: Active arcs

Grammar

S → V NP
S → V NP PP
NP → (Art) N
NP → NP PP
PP → P NP

{V, Art, N, P, N}
. . .
. . .
{S, NP, P, N}
{NP, P, N}
{P, N}

Agenda

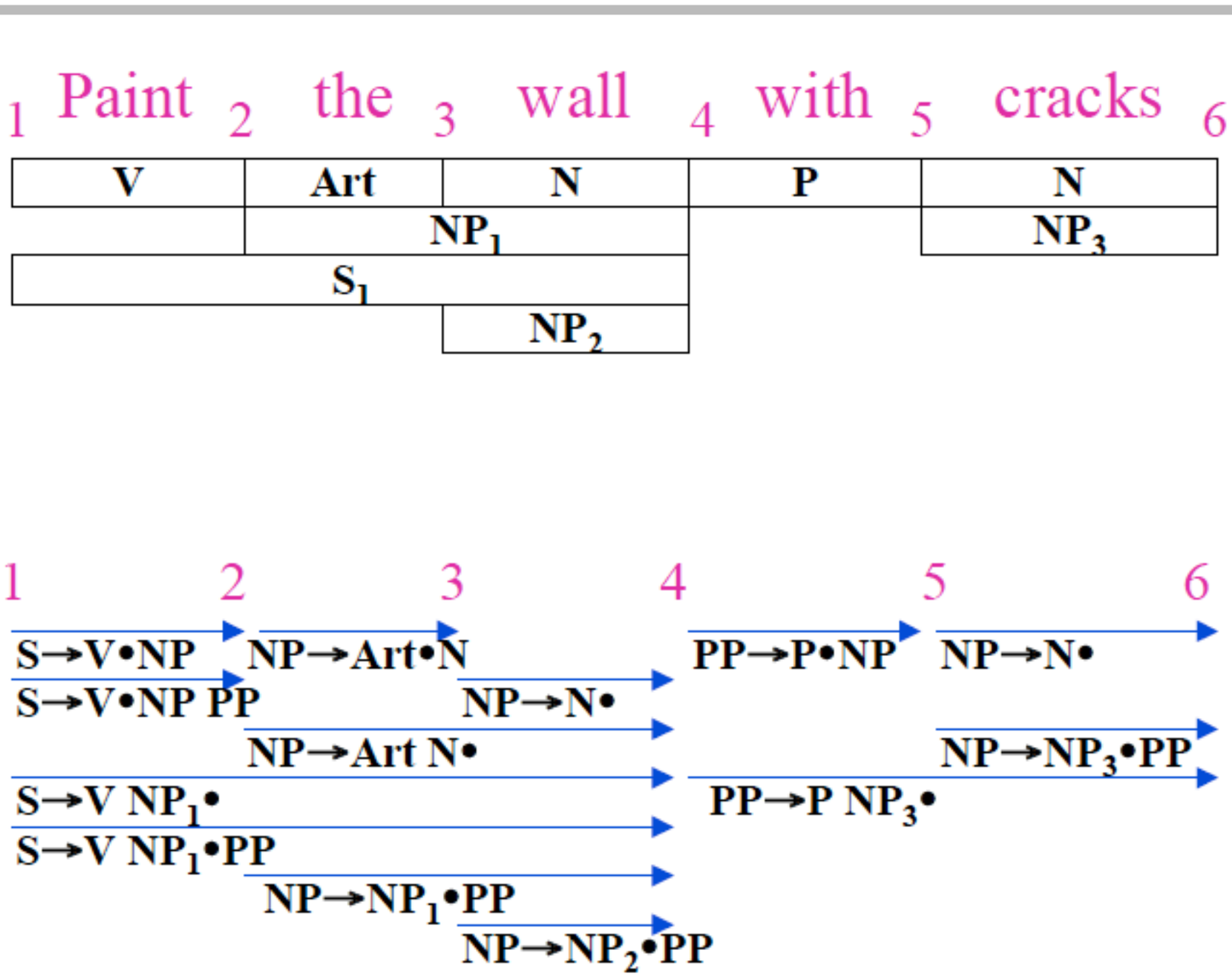



Chart: Completed arcs

Chart: Active arcs

Grammar

$S \rightarrow V\ NP$
$S \rightarrow V\ NP\ PP$
$NP \rightarrow (Art)\ N$
$NP \rightarrow NP\ PP$
$PP \rightarrow P\ NP$

{V, Art, N, P, N}

. . .

. . .

{P, N}

{N}

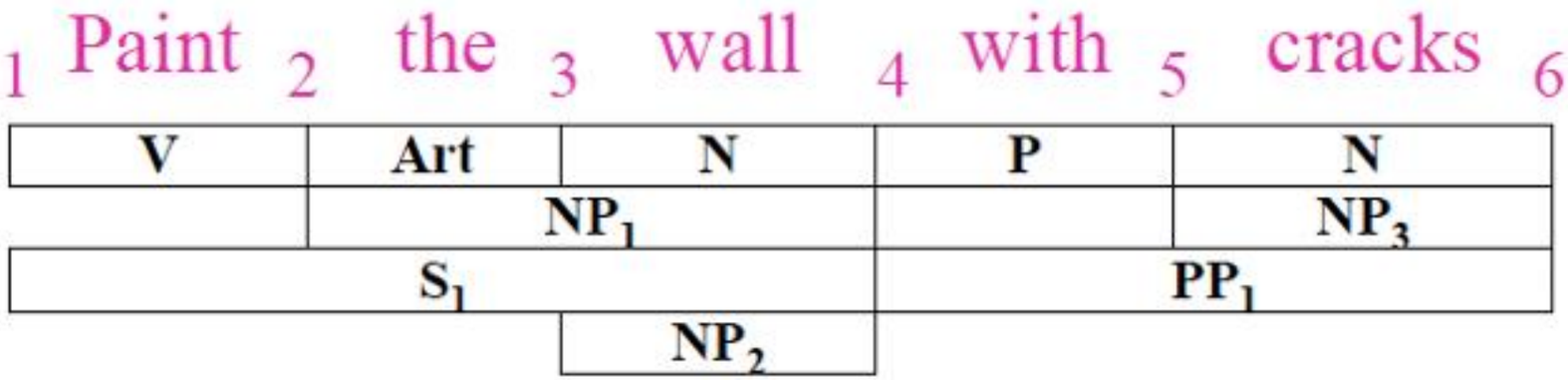
{}

{NP}

{}

{PP}

Agenda



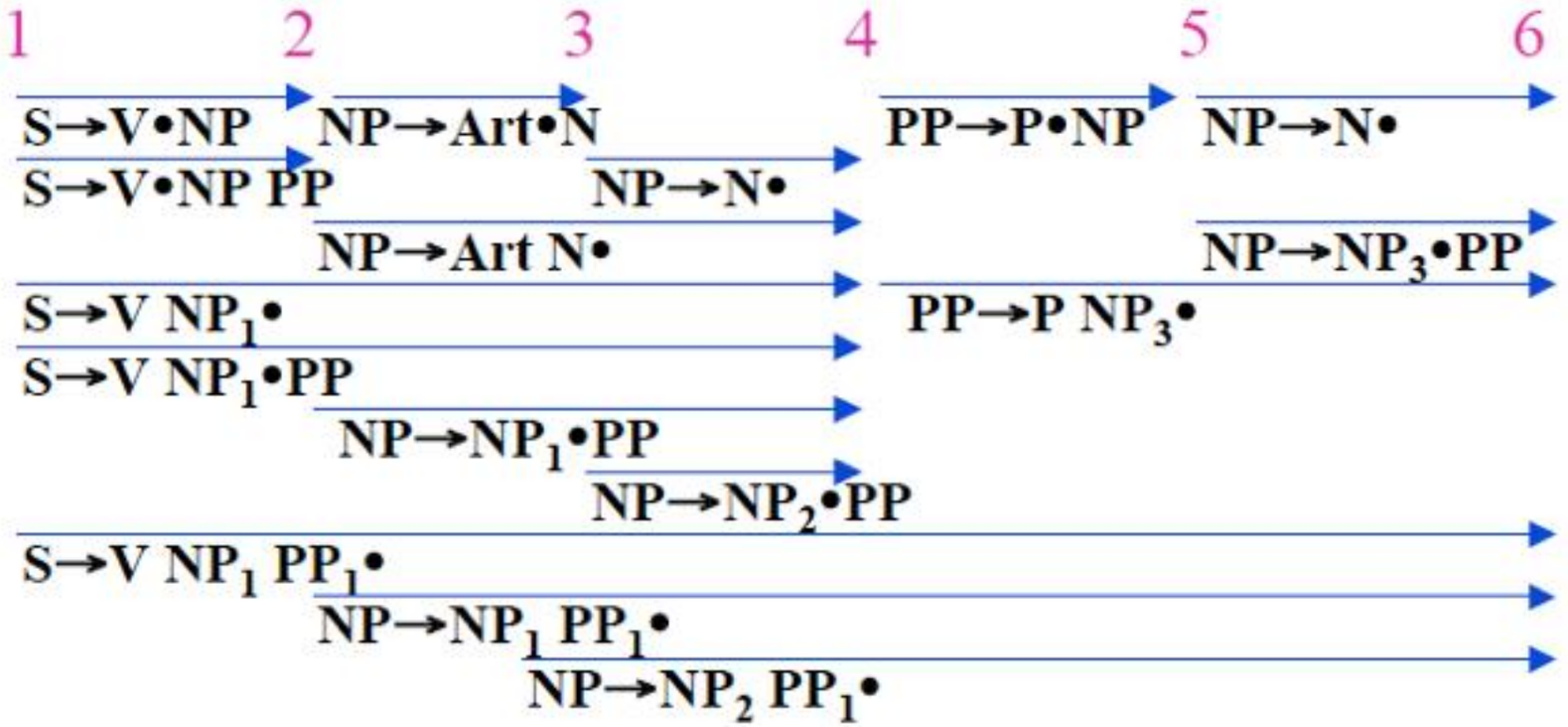




Chart: Completed arcs

Chart: Active arcs

Grammar

S → V NP
S → V NP PP
NP → (Art) N
NP → NP PP
PP → P NP

{V, Art, N, P, N}

. . .

. . .

{PP}

{}

{NP, NP, S}

Agenda

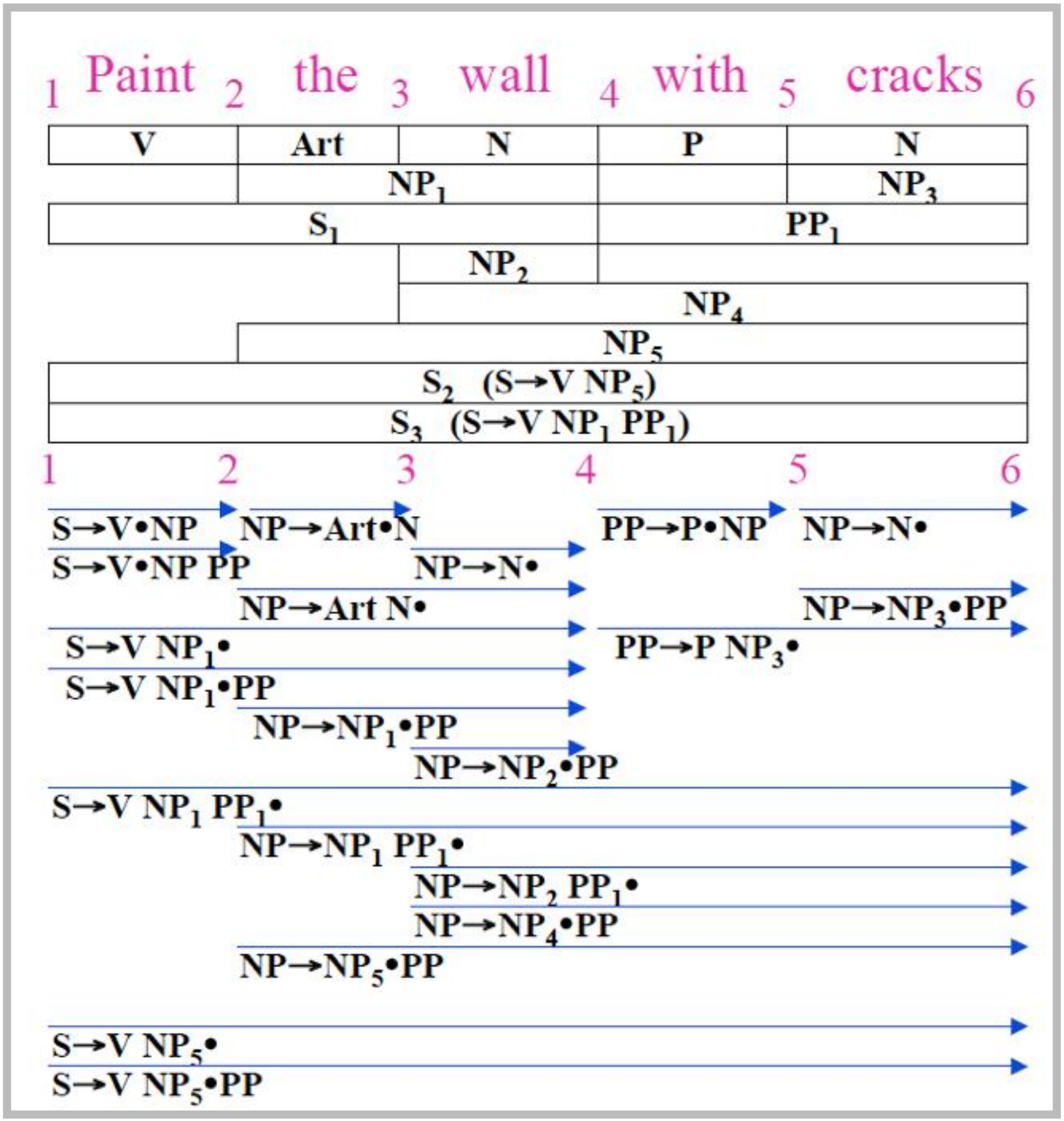



Chart: Completed arcs

Chart: Active arcs

Grammar

S → V NP
S → V NP PP
NP → (Art) N
NP → NP PP
PP → P NP

{V, Art, N, P, N}
. . .
. . .
{NP, NP, S}
{NP, S}
{S}
{S, S}
{S}
{}

Agenda