Stuttgart Fall School in CL
Class 3:

Dr. Meaghan Fowlie

September 17, 2019

Review Quiz

G; S is the start category

1. S→a S
2. S→b B
3. B→b B
4. S→ϵ
5. B→ϵ

1. Draw parse trees for *aabb* and *aaaab* using G
2. True or False?
   a. L(G) is a regular language
   b. L(G) is a context free language
   c. L(G) is a context sensitive language
   d. L(G) is a recursively enumerable language

Getting a tree

Task 1: given a grammar and a string, figure out if the string is in the language
Task 2: given a grammar and a string, give (all of the) parse tree(s) for that string

Question: How do you do these tasks?

Question: Could we write an algorithm to do these tasks?

Exercise: Sketch an algorithm that might solve Task 1 and/or Task 2

Getting a tree

Task 1: given a grammar and a string, figure out if the string is in the language
Task 2: given a grammar and a string, give (all of the) parse tree(s) for that string

Question: How do you do these tasks?

Question: Could we write an algorithm to do these tasks?

Exercise: Sketch an algorithm that might solve Task 1 and/or Task 2

Getting a tree

Task 1: given a grammar and a string, figure out if the string is in the language
Task 2: given a grammar and a string, give (all of the) parse tree(s) for that string

Question: How do you do these tasks?

Question: Could we write an algorithm to do these tasks?

Exercise: Sketch an algorithm that might solve Task 1 and/or Task 2

Getting a tree

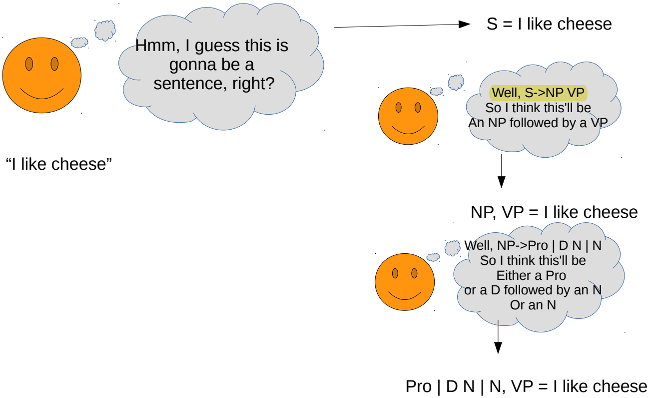
Task 1: given a grammar and a string, figure out if the string is in the language
Task 2: given a grammar and a string, give (all of the) parse tree(s) for that string

Question: How do you do these tasks?

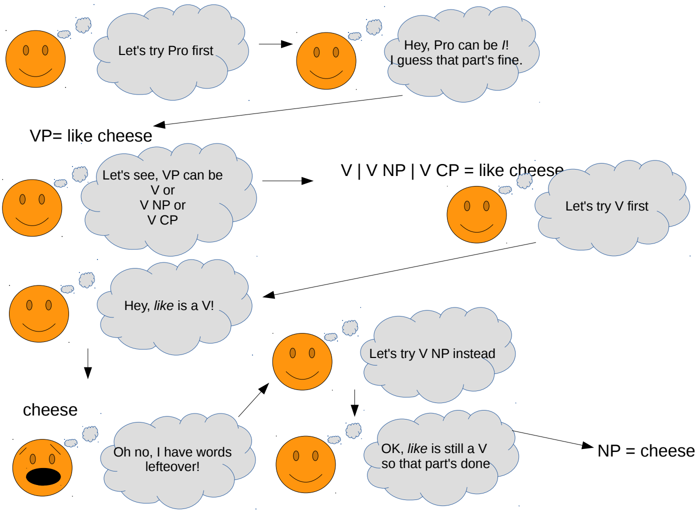
Question: Could we write an algorithm to do these tasks?

Exercise: Sketch an algorithm that might solve Task 1 and/or Task 2

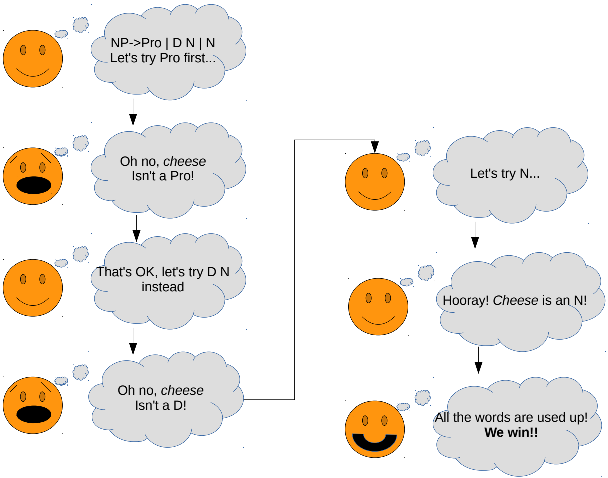
Top-Down recogniser



Top-Down recogniser



Top-Down recogniser



Top-Down recogniser

- Also called LL for **L**eft-to-right and **L**eftmost derivation
- 2 functions/inference rules
  1. `expandComplete` expands a predicted category into its right hand side
  2. `shiftComplete` removes the first word of the input and the first predicted element when they match
- **Notation:** for $s, t \in \Sigma^*$, $C, D \in V^*$, $(s, C) \vdash (t, D)$ means there's a step (expandComplete or shiftComplete) from (s, C) to (t, D).
- **Notation:** A predicted category/word C is written with a line over it: $\overline{C}$

Top-Down recogniser

- Also called LL for **L**eft-to-right and **L**eftmost derivation
- 2 functions/inference rules
  1. `expandComplete` expands a predicted category into its right hand side
  2. `shiftComplete` removes the first word of the input and the first predicted element when they match
- **Notation:** for $s, t \in \Sigma^*$, $C, D \in V^*$, $(s, C) \vdash (t, D)$ means there's a step (expandComplete or shiftComplete) from (s, C) to (t, D).
- **Notation:** A predicted category/word C is written with a line over it: $\overline{C}$

Top-Down recogniser

- Also called LL for **L**eft-to-right and **L**eftmost derivation
- 2 functions/inference rules
  1. `expandComplete` expands a predicted category into its right hand side
  2. `shiftComplete` removes the first word of the input and the first predicted element when they match
- **Notation:** for $s, t \in \Sigma^*$, $C, D \in V^*$, $(s, C) \vdash (t, D)$ means there's a step (expandComplete or shiftComplete) from (s, C) to (t, D).
- **Notation:** A predicted category/word C is written with a line over it: $\overline{C}$

Top-Down recogniser

- Also called LL for **L**eft-to-right and **L**eftmost derivation
- 2 functions/inference rules
    1. `expandComplete` expands a predicted category into its right hand side
    2. `shiftComplete` removes the first word of the input and the first predicted element when they match
- **Notation:** for $s, t \in \Sigma^*$, $C, D \in V^*$, $(s, C) \vdash (t, D)$ means there's a step (expandComplete or shiftComplete) from (s, C) to (t, D).
- **Notation:** A predicted category/word C is written with a line over it: $\overline{C}$

Top-Down recogniser

- Also called LL for **L**eft-to-right and **L**eftmost derivation
- 2 functions/inference rules
  1. `expandComplete` expands a predicted category into its right hand side
  2. `shiftComplete` removes the first word of the input and the first predicted element when they match
- **Notation:** for $s, t \in \Sigma^*$, $C, D \in V^*$, $(s, C) \vdash (t, D)$ means there's a step (expandComplete or shiftComplete) from (s, C) to (t, D).
- **Notation:** A predicted category/word C is written with a line over it: $\overline{C}$

Top-Down recogniser

- Also called LL for **L**eft-to-right and **L**eftmost derivation
- 2 functions/inference rules
    1. `expandComplete` expands a predicted category into its right hand side
    2. `shiftComplete` removes the first word of the input and the first predicted element when they match
- **Notation:** for $s, t \in \Sigma^*$, $C, D \in V^*$, $(s, C) \vdash (t, D)$ means there's a step (expandComplete or shiftComplete) from (s, C) to (t, D).
- **Notation:** A predicted category/word C is written with a line over it: $\overline{C}$

Top-Down recogniser

Notation: $f(a, b) = c$ $\qquad \dfrac{(a, b)}{c} f$

$$\frac{(\text{input}, \overline{C}\ \text{cats})}{(\text{input}, \overline{x_0 x_1} \ldots \overline{x_n}\ \text{cats})}(\texttt{expandComplete}) \qquad \text{if } C \mapsto x_0 x_1 \ldots x_n$$

$$\frac{(w\ \text{input}, \overline{w}\ \text{cats})}{(\text{input,cats})}(\texttt{shiftComplete}) \qquad \text{for } w \in \Sigma$$

If you can derive (ϵ, ϵ) from (sentence, Start), sentence is in the language.

Top-Down recogniser

Notation: $f(a,b)=c$ $\qquad \dfrac{(a,b)}{c}f$

$$\frac{(\text{input}, \overline{C}\ \text{cats})}{(\text{input}, \overline{x_0}\overline{x_1}\ldots\overline{x_n}\ \text{cats})}(\texttt{expandComplete}) \qquad \text{if } C \mapsto x_0 x_1 \ldots x_n$$

$$\frac{(w\ \text{input}, \overline{w}\ \text{cats})}{(\text{input,cats})}(\texttt{shiftComplete}) \qquad \text{for } w \in \Sigma$$

If you can derive (ϵ, ϵ) from (sentence, Start), sentence is in the language.

Top-Down recogniser

Notation: $f(a,b)=c$ $\qquad \dfrac{(a,b)}{c}f$

$$\frac{(\text{input}, \overline{C}\ \text{cats})}{(\text{input}, \overline{x_0}\overline{x_1}\ldots\overline{x_n}\ \text{cats})}(\texttt{expandComplete}) \qquad \text{if } C \mapsto x_0 x_1 \ldots x_n$$

$$\frac{(w\ \text{input}, \overline{w}\ \text{cats})}{(\text{input},\text{cats})}(\texttt{shiftComplete}) \qquad \text{for } w \in \Sigma$$

If you can derive (ϵ, ϵ) from (sentence, Start), sentence is in the language.

Top-Down recogniser

Notation: $f(a,b)=c$ $\qquad \dfrac{(a,b)}{c}f$

$$\frac{(\text{input}, \overline{C}\ \text{cats})}{(\text{input}, \overline{x_0}\overline{x_1}\ldots\overline{x_n}\ \text{cats})}(\texttt{expandComplete}) \qquad \text{if } C \mapsto x_0x_1...x_n$$

$$\frac{(w\ \text{input}, \overline{w}\ \text{cats})}{(\text{input,cats})}(\texttt{shiftComplete}) \qquad \text{for } w \in \Sigma$$

If you can derive (ϵ, ϵ) from (sentence, Start), sentence is in the language.

Top-Down recogniser

1. S$\rightarrow$a S b
2. S$\rightarrow\epsilon$

Q: Is *aabb* in *L*?

1. (aabb, $\overline{S}$) — Start
2. (aabb, $\overline{\mathbf{aSb}}$) — expandComplete (S$\rightarrow$aSb)
3. (abb, $\overline{Sb}$) — shiftComplete
4. (abb, $\overline{\mathbf{aSb}b}$) — expandComplete (S$\rightarrow$aSb)
5. (bb, $\overline{\mathbf{S}bb}$) — shiftComplete
6. (bb, $\overline{bb}$) — expandComplete (S$\rightarrow\epsilon$)
7. (b, $\overline{b}$) — shiftComplete
8. (ϵ, ϵ) — shiftComplete

Top-Down recogniser

This is “right”, in that it’s sound and complete.

- **Sound:** if you can derive (ϵ, ϵ) from (s,Start) then s is indeed in the language
- **Complete:** If s is in the language, then you can derive (ϵ, ϵ) from (s,Start)

Top-Down recogniser

This is “right”, in that it’s sound and complete.

- **Sound:** if you can derive (ϵ, ϵ) from (s,Start) then s is indeed in the language
- **Complete:** If s is in the language, then you can derive (ϵ, ϵ) from (s,Start)

A Grammar

- S→DP VP
- DP →D NP
- NP→AP NP | NP PP | N (PP) | N CP
- AP→(Adv) A
- PP →P DP
- VP →V (DP) | V CP
- CP →C S
- D→the | every | some | a
- N→idea | cat | boy | claim
- Adv →very | surprisingly
- A→good | big | silly | clever
- P→to | on | with
- V→slept | saw | thought | believed
- C →that | ϵ

Top-Down recogniser

0. (The cat slept, $\overline{S}$)
1. (The cat slept, $\overline{DP}\ \overline{VP}$)
2. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
3. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
4. (cat slept, $\overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{N}\ \overline{VP}$)
6. (cat slept, $\overline{cat}\ \overline{VP}$)
7. (slept, $\overline{VP}$)
8. (slept, $\overline{V}$)
9. (slept, $\overline{slept}$)
10. (ϵ, ϵ)

Top-Down recogniser

0. (The cat slept, $\overline{S}$)
1. (The cat slept, $\overline{DP}\ \overline{VP}$)
2. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
3. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
4. (cat slept, $\overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{N}\ \overline{VP}$)
6. (cat slept, $\overline{cat}\ \overline{VP}$)
7. (slept, $\overline{VP}$)
8. (slept, $\overline{V}$)
9. (slept, $\overline{slept}$)
10. (ϵ, ϵ)

Top-Down recogniser

0. (The cat slept, $\overline{S}$)
1. (The cat slept, $\overline{DP}\ \overline{VP}$)
2. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
3. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
4. (cat slept, $\overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{N}\ \overline{VP}$)
6. (cat slept, $\overline{cat}\ \overline{VP}$)
7. (slept, $\overline{VP}$)
8. (slept, $\overline{V}$)
9. (slept, $\overline{slept}$)
10. (ϵ, ϵ)

Top-Down recogniser

0. (The cat slept, $\overline{S}$)
1. (The cat slept, $\overline{DP}\ \overline{VP}$)
2. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
3. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
4. (cat slept, $\overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{N}\ \overline{VP}$)
6. (cat slept, $\overline{cat}\ \overline{VP}$)
7. (slept, $\overline{VP}$)
8. (slept, $\overline{V}$)
9. (slept, $\overline{slept}$)
10. (ϵ, ϵ)

Top-Down recogniser

0. (The cat slept, $\overline{S}$)
1. (The cat slept, $\overline{DP}\ \overline{VP}$)
2. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
3. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
4. (cat slept, $\overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{N}\ \overline{VP}$)
6. (cat slept, $\overline{cat}\ \overline{VP}$)
7. (slept, $\overline{VP}$)
8. (slept, $\overline{V}$)
9. (slept, $\overline{slept}$)
10. (ϵ, ϵ)

Top-Down recogniser

0. (The cat slept, $\overline{S}$)
1. (The cat slept, $\overline{DP}\ \overline{VP}$)
2. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
3. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
4. (cat slept, $\overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{N}\ \overline{VP}$)
6. (cat slept, $\overline{cat}\ \overline{VP}$)
7. (slept, $\overline{VP}$)
8. (slept, $\overline{V}$)
9. (slept, $\overline{slept}$)
10. (ϵ, ϵ)

Top-Down recogniser

0. (The cat slept, $\overline{S}$)
1. (The cat slept, $\overline{DP}$ $\overline{VP}$)
2. (The cat slept, $\overline{D}$ $\overline{NP}$ $\overline{VP}$)
3. (The cat slept, $\overline{the}$ $\overline{NP}$ $\overline{VP}$)
4. (cat slept, $\overline{NP}$ $\overline{VP}$)
5. (cat slept, $\overline{N}$ $\overline{VP}$)
6. (cat slept, $\overline{cat}$ $\overline{VP}$)
7. (slept, $\overline{VP}$)
8. (slept, $\overline{V}$)
9. (slept, $\overline{slept}$)
10. (ϵ, ϵ)

Top-Down recogniser

0. (The cat slept, $\overline{S}$)
1. (The cat slept, $\overline{DP}$ $\overline{VP}$)
2. (The cat slept, $\overline{D}$ $\overline{NP}$ $\overline{VP}$)
3. (The cat slept, $\overline{the}$ $\overline{NP}$ $\overline{VP}$)
4. (cat slept, $\overline{NP}$ $\overline{VP}$)
5. (cat slept, $\overline{N}$ $\overline{VP}$)
6. (cat slept, $\overline{cat}$ $\overline{VP}$)
7. (slept, $\overline{VP}$)
8. (slept, $\overline{V}$)
9. (slept, $\overline{slept}$)
10. (ϵ, ϵ)

Top-Down recogniser

0. (The cat slept, $\overline{S}$)
1. (The cat slept, $\overline{DP}\ \overline{VP}$)
2. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
3. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
4. (cat slept, $\overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{N}\ \overline{VP}$)
6. (cat slept, $\overline{cat}\ \overline{VP}$)
7. (slept, $\overline{VP}$)
8. (slept, $\overline{V}$)
9. (slept, $\overline{slept}$)
10. (ϵ, ϵ)

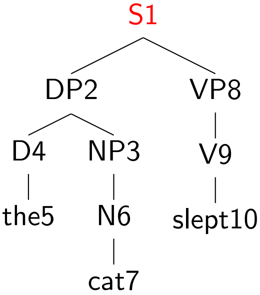
Top-Down recogniser

0. (The cat slept, $\overline{S}$)
1. (The cat slept, $\overline{DP}\ \overline{VP}$)
2. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
3. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
4. (cat slept, $\overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{N}\ \overline{VP}$)
6. (cat slept, $\overline{cat}\ \overline{VP}$)
7. (slept, $\overline{VP}$)
8. (slept, $\overline{V}$)
9. (slept, $\overline{slept}$)
10. (ϵ, ϵ)

Top-Down recogniser

0. (The cat slept, $\overline{S}$)
1. (The cat slept, $\overline{DP}\ \overline{VP}$)
2. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
3. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
4. (cat slept, $\overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{N}\ \overline{VP}$)
6. (cat slept, $\overline{cat}\ \overline{VP}$)
7. (slept, $\overline{VP}$)
8. (slept, $\overline{V}$)
9. (slept, $\overline{slept}$)
10. (ϵ, ϵ)

Top-Down recogniser



Top-Down recogniser

Query: How do we chose the right expansion of a rule?

Top-Down recogniser + backtracking

Q: How do we chose the right expansion of a rule?

A: Try the first one, but give ourselves the possibility of backing up and trying the next one, by recording all expansions.

Write down all the expansions of pair of (input, predictions):

$$\frac{\mathit{pair} :: \mathit{pairs}}{p_0 p_1 \ldots p_n :: \mathit{pairs}}(\texttt{step}) \qquad \text{if } \mathit{pair} \vdash \{p_0, p_1, \ldots p_n\}$$

If we're stuck, go on to the next guess:

$$\frac{\mathit{pair} :: \mathit{pairs}}{\mathit{pairs}}(\texttt{backtrack}) \qquad \text{if } \mathit{pair} \vdash \emptyset$$

Top-Down recogniser + backtracking

Q: How do we chose the right expansion of a rule?

A: Try the first one, but give ourselves the possibility of backing up and trying the next one, by recording all expansions.

Write down all the expansions of pair of (input, predictions):

$$\frac{\mathit{pair} :: \mathit{pairs}}{p_0 p_1 \dots p_n :: \mathit{pairs}}(\texttt{step}) \qquad \text{if } \mathit{pair} \vdash \{p_0, p_1, \dots p_n\}$$

If we're stuck, go on to the next guess:

$$\frac{\mathit{pair} :: \mathit{pairs}}{\mathit{pairs}}(\texttt{backtrack}) \qquad \text{if } \mathit{pair} \vdash \emptyset$$

Top-Down recogniser + backtracking

Q: How do we chose the right expansion of a rule?

A: Try the first one, but give ourselves the possibility of backing up and trying the next one, by recording all expansions.

Write down all the expansions of pair of (input, predictions):

$$\frac{pair :: pairs}{p_0 p_1 \ldots p_n :: pairs}(\texttt{step}) \qquad \text{if } pair \vdash \{p_0, p_1, \ldots p_n\}$$

If we're stuck, go on to the next guess:

$$\frac{pair :: pairs}{pairs}(\texttt{backtrack}) \qquad \text{if } pair \vdash \emptyset$$

Top-Down recogniser + backtracking

Q: How do we chose the right expansion of a rule?

A: Try the first one, but give ourselves the possibility of backing up and trying the next one, by recording all expansions.

Write down all the expansions of pair of (input, predictions):

$$\frac{\mathit{pair} :: \mathit{pairs}}{p_0 p_1 \ldots p_n :: \mathit{pairs}}(\texttt{step}) \qquad \text{if } \mathit{pair} \vdash \{p_0, p_1, \ldots p_n\}$$

If we're stuck, go on to the next guess:

$$\frac{\mathit{pair} :: \mathit{pairs}}{\mathit{pairs}}(\texttt{backtrack}) \qquad \text{if } \mathit{pair} \vdash \emptyset$$

Top-Down recogniser + backtracking

Q: How do we chose the right expansion of a rule?

A: Try the first one, but give ourselves the possibility of backing up and trying the next one, by recording all expansions.

Write down all the expansions of pair of (input, predictions):

$$\frac{\mathit{pair} :: \mathit{pairs}}{p_0 p_1 \ldots p_n :: \mathit{pairs}}(\mathtt{step}) \qquad \text{if } \mathit{pair} \vdash \{p_0, p_1, \ldots p_n\}$$

If we're stuck, go on to the next guess:

$$\frac{\mathit{pair} :: \mathit{pairs}}{\mathit{pairs}}(\mathtt{backtrack}) \qquad \text{if } \mathit{pair} \vdash \emptyset$$

Top-Down recogniser + backtracking

Q: How do we chose the right expansion of a rule?

A: Try the first one, but give ourselves the possibility of backing up and trying the next one, by recording all expansions.

Write down all the expansions of pair of (input, predictions):

$$\frac{\mathit{pair} :: \mathit{pairs}}{p_0 p_1 \ldots p_n :: \mathit{pairs}}(\texttt{step}) \qquad \text{if } \mathit{pair} \vdash \{p_0, p_1, \ldots p_n\}$$

If we're stuck, go on to the next guess:

$$\frac{\mathit{pair} :: \mathit{pairs}}{\mathit{pairs}}(\texttt{backtrack}) \qquad \text{if } \mathit{pair} \vdash \emptyset$$

Top-Down recogniser + backtracking

Q: How do we chose the right expansion of a rule?

A: Try the first one, but give ourselves the possibility of backing up and trying the next one, by recording all expansions.

Write down all the expansions of pair of (input, predictions):

$$\frac{\mathit{pair} :: \mathit{pairs}}{p_0 p_1 \dots p_n :: \mathit{pairs}}(\texttt{step}) \qquad \text{if } \mathit{pair} \vdash \{p_0, p_1, \dots p_n\}$$

If we're stuck, go on to the next guess:

$$\frac{\mathit{pair} :: \mathit{pairs}}{\mathit{pairs}}(\texttt{backtrack}) \qquad \text{if } \mathit{pair} \vdash \emptyset$$

A Grammar

- S→DP VP
- DP →D NP
- NP→A NP | N | N PP
- PP →P DP
- VP →V | V DP | V CP
- CP →C S
- D→the
- N→idea | cat | claim
- A→good | big
- P→to
- V→slept | saw
- C →that

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}\ \overline{VP}$)
3. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
4. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{NP}\ \overline{VP}$)
6.
   - (cat slept, $\overline{A}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
7.
   - (cat slept, $\overline{good}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}$ $\overline{VP}$)
3. (The cat slept, $\overline{D}$ $\overline{NP}$ $\overline{VP}$)
4. (The cat slept, $\overline{the}$ $\overline{NP}$ $\overline{VP}$)
5. (cat slept, $\overline{NP}$ $\overline{VP}$)
6. - (cat slept, $\overline{A}$ $\overline{NP}$ $\overline{VP}$)
   - (cat slept, $\overline{N}$ $\overline{VP}$)
   - (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)
7. - (cat slept, $\overline{good}$ $\overline{NP}$ $\overline{VP}$)
   - (cat slept, $\overline{big}$ $\overline{NP}$ $\overline{VP}$)
   - (cat slept, $\overline{N}$ $\overline{VP}$)
   - (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}\ \overline{VP}$)
3. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
4. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{NP}\ \overline{VP}$)
6.
   - (cat slept, $\overline{A}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
7.
   - (cat slept, $\overline{good}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}\ \overline{VP}$)
3. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
4. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{NP}\ \overline{VP}$)
6.
   - (cat slept, $\overline{A}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
7.
   - (cat slept, $\overline{good}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}\ \overline{VP}$)
3. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
4. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{NP}\ \overline{VP}$)
6.
   - (cat slept, $\overline{A}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
7.
   - (cat slept, $\overline{good}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}\ \overline{VP}$)
3. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
4. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{NP}\ \overline{VP}$)
6.
   - (cat slept, $\overline{A}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
7.
   - (cat slept, $\overline{good}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}\ \overline{VP}$)
3. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
4. (The cat slept, $\overline{\textit{the}}\ \overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{NP}\ \overline{VP}$)
6.
   - (cat slept, $\overline{A}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
7.
   - (cat slept, $\overline{\textit{good}}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{\textit{big}}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}\ \overline{VP}$)
3. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
4. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{NP}\ \overline{VP}$)
6.
   - (cat slept, $\overline{A}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
7.
   - (cat slept, $\overline{good}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}\ \overline{VP}$)
3. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
4. (The cat slept, $\overline{\textit{the}}\ \overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{NP}\ \overline{VP}$)
6.
   - (cat slept, $\overline{A}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
7.
   - (cat slept, $\overline{\textit{good}}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{\textit{big}}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}\ \overline{VP}$)
3. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
4. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{NP}\ \overline{VP}$)
6.
   - (cat slept, $\overline{A}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
7.
   - (cat slept, $\overline{good}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}\ \overline{VP}$)
3. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
4. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{NP}\ \overline{VP}$)
6.
   - (cat slept, $\overline{A}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
7.
   - (cat slept, $\overline{good}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}\ \overline{VP}$)
3. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
4. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{NP}\ \overline{VP}$)
6.
   - (cat slept, $\overline{A}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
7.
   - (cat slept, $\overline{good}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}\ \overline{VP}$)
3. (The cat slept, $\overline{D}\ \overline{NP}\ \overline{VP}$)
4. (The cat slept, $\overline{the}\ \overline{NP}\ \overline{VP}$)
5. (cat slept, $\overline{NP}\ \overline{VP}$)
6.
   - (cat slept, $\overline{A}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
7.
   - (cat slept, $\overline{good}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

1. (The cat slept, $\overline{S}$)
2. (The cat slept, $\overline{DP}$ $\overline{VP}$)
3. (The cat slept, $\overline{D}$ $\overline{NP}$ $\overline{VP}$)
4. (The cat slept, $\overline{the}$ $\overline{NP}$ $\overline{VP}$)
5. (cat slept, $\overline{NP}$ $\overline{VP}$)
6. - (cat slept, $\overline{A}$ $\overline{NP}$ $\overline{VP}$)
   - (cat slept, $\overline{N}$ $\overline{VP}$)
   - (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)
7. - (cat slept, $\overline{good}$ $\overline{NP}$ $\overline{VP}$)
   - (cat slept, $\overline{big}$ $\overline{NP}$ $\overline{VP}$)
   - (cat slept, $\overline{N}$ $\overline{VP}$)
   - (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)

Top-Down recogniser + backtracking

8.
- (cat slept, $\overline{big}$ $\overline{NP}$ $\overline{VP}$)
- (cat slept, $\overline{N}$ $\overline{VP}$)
- (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)

9.
- (cat slept, $\overline{N}$ $\overline{VP}$)
- (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)

10.
- (cat slept, $\overline{idea}$ $\overline{VP}$)
- (cat slept, $\overline{cat}$ $\overline{VP}$)
- (cat slept, $\overline{claim}$ $\overline{VP}$)
- (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)

11.
- (cat slept, $\overline{cat}$ $\overline{VP}$)
- (cat slept, $\overline{claim}$ $\overline{VP}$)
- (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)

12.
- (slept, $\overline{VP}$)
- (cat slept, $\overline{claim}$ $\overline{VP}$)
- (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)

Top-Down recogniser + backtracking

8.
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
9.
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
10.
   - (cat slept, $\overline{idea}\ \overline{VP}$)
   - (cat slept, $\overline{cat}\ \overline{VP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
11.
   - (cat slept, $\overline{cat}\ \overline{VP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
12.
   - (slept, $\overline{VP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

8.
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
9.
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
10.
    - (cat slept, $\overline{idea}\ \overline{VP}$)
    - (cat slept, $\overline{cat}\ \overline{VP}$)
    - (cat slept, $\overline{claim}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
11.
    - (cat slept, $\overline{cat}\ \overline{VP}$)
    - (cat slept, $\overline{claim}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
12.
    - (slept, $\overline{VP}$)
    - (cat slept, $\overline{claim}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

8.
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
9.
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
10.
   - (cat slept, $\overline{idea}\ \overline{VP}$)
   - (cat slept, $\overline{cat}\ \overline{VP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
11.
   - (cat slept, $\overline{cat}\ \overline{VP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
12.
   - (slept, $\overline{VP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

8.
   - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
9.
   - (cat slept, $\overline{N}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
10.
   - (cat slept, $\overline{idea}\ \overline{VP}$)
   - (cat slept, $\overline{cat}\ \overline{VP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
11.
   - (cat slept, $\overline{cat}\ \overline{VP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
12.
   - (slept, $\overline{VP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

8.
    - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
9.
    - (cat slept, $\overline{N}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
10.
    - (cat slept, $\overline{idea}\ \overline{VP}$)
    - (cat slept, $\overline{cat}\ \overline{VP}$)
    - (cat slept, $\overline{claim}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
11.
    - (cat slept, $\overline{cat}\ \overline{VP}$)
    - (cat slept, $\overline{claim}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
12.
    - (slept, $\overline{VP}$)
    - (cat slept, $\overline{claim}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

8.
- (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

9.
- (cat slept, $\overline{N}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

10.
- (cat slept, $\overline{idea}\ \overline{VP}$)
- (cat slept, $\overline{cat}\ \overline{VP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

11.
- (cat slept, $\overline{cat}\ \overline{VP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

12.
- (slept, $\overline{VP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

8.
    - (cat slept, $\overline{big}\ \overline{NP}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
9.
    - (cat slept, $\overline{N}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
10.
    - (cat slept, $\overline{idea}\ \overline{VP}$)
    - (cat slept, $\overline{cat}\ \overline{VP}$)
    - (cat slept, $\overline{claim}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
11.
    - (cat slept, $\overline{cat}\ \overline{VP}$)
    - (cat slept, $\overline{claim}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
12.
    - (slept, $\overline{VP}$)
    - (cat slept, $\overline{claim}\ \overline{VP}$)
    - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

13.
   - (slept, $\overline{V}$)
   - (slept, $\overline{V}\ \overline{DP}$)
   - (slept, $\overline{V}\ \overline{CP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
14.
   - (slept, $\overline{slept}$)
   - (slept, $\overline{saw}$)
   - (slept, $\overline{V}\ \overline{DP}$)
   - (slept, $\overline{V}\ \overline{CP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
15.
   - (ϵ, ϵ)
   - (slept, $\overline{saw}$)
   - (slept, $\overline{V}\ \overline{DP}$)
   - (slept, $\overline{V}\ \overline{CP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

13
- (slept, $\overline{V}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

14
- (slept, $\overline{slept}$)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

15
- (ϵ, ϵ)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

13.
- (slept, $\overline{V}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

14.
- (slept, $\overline{slept}$)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

15.
- (ϵ, ϵ)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

13.
- (slept, $\overline{V}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

14.
- (slept, $\overline{slept}$)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

15.
- (ϵ, ϵ)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

13.
   - (slept, $\overline{V}$)
   - (slept, $\overline{V}\ \overline{DP}$)
   - (slept, $\overline{V}\ \overline{CP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
14.
   - (slept, $\overline{slept}$)
   - (slept, $\overline{saw}$)
   - (slept, $\overline{V}\ \overline{DP}$)
   - (slept, $\overline{V}\ \overline{CP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
15.
   - (ϵ, ϵ)
   - (slept, $\overline{saw}$)
   - (slept, $\overline{V}\ \overline{DP}$)
   - (slept, $\overline{V}\ \overline{CP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

13
- (slept, $\overline{V}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

14
- (slept, $\overline{slept}$)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

15
- (ϵ, ϵ)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

13.
- (slept, $\overline{V}$)
- (slept, $\overline{V}$ $\overline{DP}$)
- (slept, $\overline{V}$ $\overline{CP}$)
- (cat slept, $\overline{claim}$ $\overline{VP}$)
- (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)

14.
- (slept, $\overline{slept}$)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}$ $\overline{DP}$)
- (slept, $\overline{V}$ $\overline{CP}$)
- (cat slept, $\overline{claim}$ $\overline{VP}$)
- (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)

15.
- (ϵ, ϵ)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}$ $\overline{DP}$)
- (slept, $\overline{V}$ $\overline{CP}$)
- (cat slept, $\overline{claim}$ $\overline{VP}$)
- (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)

Top-Down recogniser + backtracking

13.
- (slept, $\overline{V}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

14.
- (slept, $\overline{slept}$)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

15.
- (ϵ, ϵ)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}\ \overline{DP}$)
- (slept, $\overline{V}\ \overline{CP}$)
- (cat slept, $\overline{claim}\ \overline{VP}$)
- (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

13.
   - (slept, $\overline{V}$)
   - (slept, $\overline{V}\ \overline{DP}$)
   - (slept, $\overline{V}\ \overline{CP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
14.
   - (slept, $\overline{slept}$)
   - (slept, $\overline{saw}$)
   - (slept, $\overline{V}\ \overline{DP}$)
   - (slept, $\overline{V}\ \overline{CP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)
15.
   - (ϵ, ϵ)
   - (slept, $\overline{saw}$)
   - (slept, $\overline{V}\ \overline{DP}$)
   - (slept, $\overline{V}\ \overline{CP}$)
   - (cat slept, $\overline{claim}\ \overline{VP}$)
   - (cat slept, $\overline{N}\ \overline{PP}\ \overline{VP}$)

Top-Down recogniser + backtracking

13.
- (slept, $\overline{V}$)
- (slept, $\overline{V}$ $\overline{DP}$)
- (slept, $\overline{V}$ $\overline{CP}$)
- (cat slept, $\overline{claim}$ $\overline{VP}$)
- (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)

14.
- (slept, $\overline{slept}$)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}$ $\overline{DP}$)
- (slept, $\overline{V}$ $\overline{CP}$)
- (cat slept, $\overline{claim}$ $\overline{VP}$)
- (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)

15.
- (ϵ, ϵ)
- (slept, $\overline{saw}$)
- (slept, $\overline{V}$ $\overline{DP}$)
- (slept, $\overline{V}$ $\overline{CP}$)
- (cat slept, $\overline{claim}$ $\overline{VP}$)
- (cat slept, $\overline{N}$ $\overline{PP}$ $\overline{VP}$)

Exercise

G_{Eng}

$$
\begin{array}{rcl}
S & \to & DP\ VP \\
DP & \to & D\ NP \\
NP & \to & A\ NP \mid N \\
VP & \to & V \mid V\ DP \\
D & \to & \text{the} \\
N & \to & \text{idea} \mid \text{cat} \mid \text{claim} \\
A & \to & \text{good} \mid \text{big} \\
V & \to & \text{slept} \mid \text{saw}
\end{array}
$$

G_{ab}

$$
\begin{array}{rcl}
S & \to & S\ S \\
S & \to & a \mid b
\end{array}
$$

1. Use the top-down recogniser to check if G_{Eng} generates these:
    1. the idea slept the big claim
    2. cat
    3. (a sentence of your own making)
2. Use the top-down recogniser to check if G_{ab} generates these:
    1. aabb
    2. b

Discussion

What did you notice about the top-down parser?

CKY recogniser

$$\begin{array}{rcl} S & \to & DP\ VP \\ DP & \to & D\ NP \\ D & \to & the \mid every \\ N & \to & cat \mid dog \\ VP & \to & slept \mid V\ DP \\ V & \to & saw \end{array}$$

(1) $_0$ the $_1$ cat $_2$ slept $_3$

	1	2	3
0	D	DP	S
1		N	
2			VP

CKY recogniser

$$
\begin{array}{rcl}
\text{S} & \rightarrow & \text{DP VP} \\
\text{DP} & \rightarrow & \text{D NP} \\
\text{D} & \rightarrow & \text{the} \mid \text{every} \\
\text{N} & \rightarrow & \text{cat} \mid \text{dog} \\
\text{VP} & \rightarrow & \text{slept} \mid \text{V DP} \\
\text{V} & \rightarrow & \text{saw}
\end{array}
$$

(1) $_0$ the $_1$ cat $_2$ slept $_3$

	1	2	3
0	D	DP	S
1		N	
2			VP

CKY recogniser

$$\begin{array}{rcl} S & \rightarrow & DP\ VP \\ DP & \rightarrow & D\ NP \\ D & \rightarrow & \text{the} \mid \text{every} \\ N & \rightarrow & \text{cat} \mid \text{dog} \\ VP & \rightarrow & \text{slept} \mid V\ DP \\ V & \rightarrow & \text{saw} \end{array}$$

(1) $_0$ the $_1$ cat $_2$ slept $_3$

	1	2	3
0	D	DP	S
1		N	
2			VP

CKY recogniser

$$\begin{array}{rcl} S & \to & DP\ VP \\ DP & \to & D\ NP \\ D & \to & \text{the} \mid \text{every} \\ N & \to & \text{cat} \mid \text{dog} \\ VP & \to & \text{slept} \mid V\ DP \\ V & \to & \text{saw} \end{array}$$

(1) $_0$ the $_1$ cat $_2$ slept $_3$

	1	2	3
0	D	DP	S
1		N	
2			VP

CKY recogniser

$$\begin{array}{rcl} S & \rightarrow & DP\ VP \\ DP & \rightarrow & D\ NP \\ D & \rightarrow & the \mid every \\ N & \rightarrow & cat \mid dog \\ VP & \rightarrow & slept \mid V\ DP \\ V & \rightarrow & saw \end{array}$$

(1) $_0$ the $_1$ cat $_2$ slept $_3$

	1	2	3
0	D	DP	S
1		N	
2			VP

CKY

- Cocke, Kasami, and Younger
- aka CYK parsing
- a type of chart parsing
- sound and complete (Shieber et al., 1995)
- for sentence length n, maximum number of steps is proportional to n^3 (Aho and Ullman, 1972)
- Efficient enough? Disagreement in the literature.

CKY

- Cocke, Kasami, and Younger
- aka CYK parsing
- a type of chart parsing
- sound and complete (Shieber et al., 1995)
- for sentence length n, maximum number of steps is proportional to n^3 (Aho and Ullman, 1972)
- Efficient enough? Disagreement in the literature.

CKY

- Cocke, Kasami, and Younger
- aka CYK parsing
- a type of chart parsing
- sound and complete (Shieber et al., 1995)
- for sentence length n, maximum number of steps is proportional to n^3 (Aho and Ullman, 1972)
- Efficient enough? Disagreement in the literature.

CKY

- Cocke, Kasami, and Younger
- aka CYK parsing
- a type of chart parsing
- sound and complete (Shieber et al., 1995)
- for sentence length n, maximum number of steps is proportional to n^3 (Aho and Ullman, 1972)
- Efficient enough? Disagreement in the literature.

CKY

- Cocke, Kasami, and Younger
- aka CYK parsing
- a type of chart parsing
- sound and complete (Shieber et al., 1995)
- for sentence length n, maximum number of steps is proportional to n^3 (Aho and Ullman, 1972)
- Efficient enough? Disagreement in the literature.

CKY

- Cocke, Kasami, and Younger
- aka CYK parsing
- a type of chart parsing
- sound and complete (Shieber et al., 1995)
- for sentence length n, maximum number of steps is proportional to n^3 (Aho and Ullman, 1972)
- Efficient enough? Disagreement in the literature.

CKY

For string $s = w_0 w_1 \ldots w_n$ and for $i, j, k \leq n$, we use the following rules:

- $(i-1, i) : w_i$ (AXIOMS)
- $\dfrac{(i,j) : w}{(i,j)\ :\ A}$(REDUCE1) if A→w
- $\dfrac{(i,j) : B \quad (j,k) : C}{(i,k)\ :\ A}$(REDUCE2) if A→B C

$s \in L$ iff the closure of the axioms under the inference rules is $(0, n) : S$

CKY

For string $s = w_0 w_1 ... w_n$ and for $i, j, k \leq n$, we use the following rules:

- $(i-1, i) : w_i$ $(\textsc{axioms})$
- $\dfrac{(i,j) : w}{(i,j)\ :\ A}(\textsc{reduce1})$ if A→w
- $\dfrac{(i,j) : B \quad (j,k) : C}{(i,k)\ :\ A}(\textsc{reduce2})$ if A→B C

$s \in L$ iff the closure of the axioms under the inference rules is $(0, n) : S$

CKY

For string $s = w_0 w_1 ... w_n$ and for $i, j, k \leq n$, we use the following rules:

- $(i-1, i) : w_i$ (AXIOMS)
- $\dfrac{(i,j) : w}{(i,j)\ :\ A}$(REDUCE1) if A→w
- $\dfrac{(i,j) : B \quad (j,k) : C}{(i,k)\ :\ A}$(REDUCE2) if A→B C

$s \in L$ iff the closure of the axioms under the inference rules is $(0, n) : S$

CKY

For string $s = w_0 w_1 ... w_n$ and for $i, j, k \leq n$, we use the following rules:

- $(i-1, i) : w_i$ $(\textsc{axioms})$
- $\dfrac{(i,j) : w}{(i,j) \ : \ A}(\textsc{reduce1})$ if A→w
- $\dfrac{(i,j) : B \quad (j,k) : C}{(i,k) \ : \ A}(\textsc{reduce2})$ if A→B C

$s \in L$ iff the closure of the axioms under the inference rules is $(0, n) : S$

CKY

For string $s = w_0 w_1 ... w_n$ and for $i, j, k \leq n$, we use the following rules:

- $(i-1, i) : w_i$ $(\textsc{axioms})$
- $\dfrac{(i,j) : w}{(i,j) \ : \ A}(\textsc{reduce1})$ if A→w
- $\dfrac{(i,j) : B \quad (j,k) : C}{(i,k) \ : \ A}(\textsc{reduce2})$ if A→B C

$s \in L$ iff the closure of the axioms under the inference rules is $(0, n) : S$

CKY

- Fill in the chart in every way possible
- Top right corner has start category: grammatical
- relative efficiency comes from the fact that ambiguities get merged whereever possible

CKY

- Fill in the chart in every way possible
- Top right corner has start category: grammatical
- relative efficiency comes from the fact that ambiguities get merged whereever possible

CKY

- Fill in the chart in every way possible
- Top right corner has start category: grammatical
- relative efficiency comes from the fact that ambiguities get merged whereever possible

CKY

- S→S S
- S→a

(2) a. a a a a

b. $_0$ a $_1$ a $_2$ a $_3$ a $_4$

	1	2	3	4
0	S	S	S	S
1		S	S	S
2			S	S
3				S

CKY

- $S \rightarrow S\ S$
- $S \rightarrow a$

(2) a. a a a a

b. $_0$ a $_1$ a $_2$ a $_3$ a $_4$

	1	2	3	4
0	S	S	S	S
1		S	S	S
2			S	S
3				S

CKY

- S→S S
- S→a

(2) a. a a a a

b. $_0$ a $_1$ a $_2$ a $_3$ a $_4$

	1	2	3	4
0	S	S	S	S
1		S	S	S
2			S	S
3				S

CKY

- S→S S
- S→a

(2) a. a a a a

b. $_0$ a $_1$ a $_2$ a $_3$ a $_4$

	1	2	3	4
0	S	S	S	S
1		S	S	S
2			S	S
3				S

CKY

- S→S S
- S→a

(2) a. a a a a

b. $_0$ a $_1$ a $_2$ a $_3$ a $_4$

	1	2	3	4
0	S	S	S	S
1		S	S	S
2			S	S
3				S

CKY

- S→S S
- S→a

(2) a. a a a a

b. $_0$ a $_1$ a $_2$ a $_3$ a $_4$

	1	2	3	4
0	S	S	S	S
1		S	S	S
2			S	S
3				S

CKY

- S→S S
- S→a

(2) a. a a a a

b. $_0$ a $_1$ a $_2$ a $_3$ a $_4$

	1	2	3	4
0	S	S	S	S
1		S	S	S
2			S	S
3				S

CKY

- S→S S
- S→a

(2) a. a a a a

b. $_0$ a $_1$ a $_2$ a $_3$ a $_4$

	1	2	3	4
0	S	S	S	S
1		S	S	S
2			S	S
3				S

CKY

- S→S S
- S→a

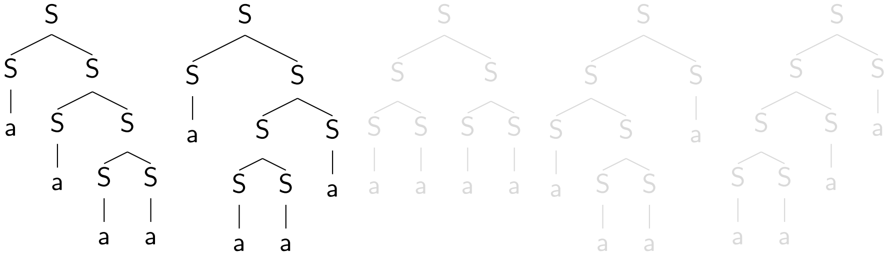
(2) a. a a a a

b. $_0$ a $_1$ a $_2$ a $_3$ a $_4$

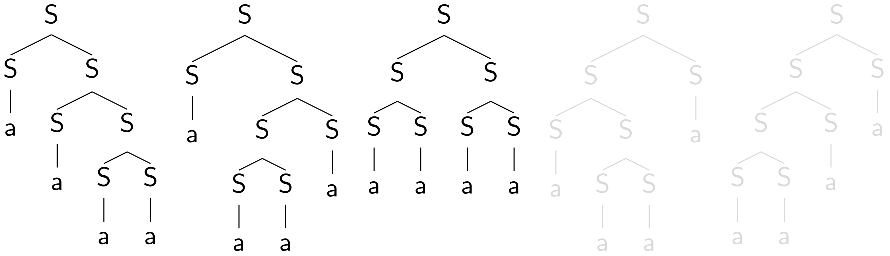
	1	2	3	4
0	S	S	S	S
1		S	S	S
2			S	S
3				S

CKY

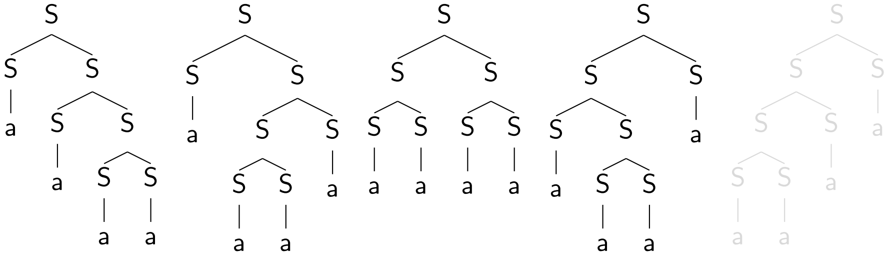
CKY



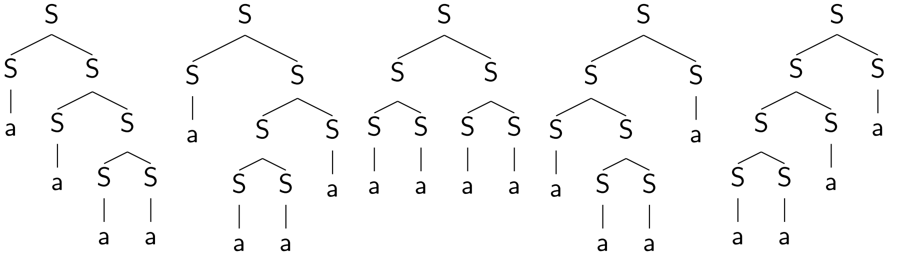
CKY



CKY



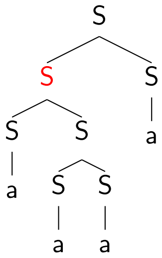
CKY

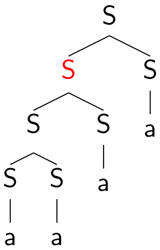


CKY

	1	2	3	4
0	S	S	S	S
1		S	S	S
2			S	S
3				S

CKY

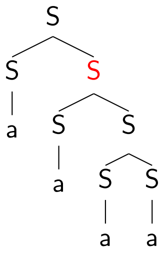


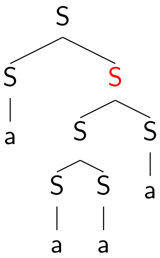


	1	2	3	4
0	S	S	S	S
1		S	S	S
2			S	S
3				S

CKY

	1	2	3	4
0	S	S	S	S
1		S	S	S
2			S	S
3				S





CKY

	1	2	3	4
0	S	S	S	S
1		S	S	S
2			S	S
3				S

CKY practice

- S→DP VP
- DP→D NP
- D→the, a
- NP→man, woman, cat, telescope, NP PP
- PP→P DP
- P→with, on, to
- VP→slept, fell, V DP, VP PP
- V→saw, hit

(3) the woman with the cat fell

(4) the man saw the woman with the telescope

(5) the woman the cat with

Chomsky normal form

Traditionally, CKY parsers are defined over grammars in Chomsky Normal Form:

Definition

G is in CNF iff all production rules have one of the following forms:

- A→x $\quad A \in Cat, x \in \Sigma$
- A→B C $\quad A, B, C \in Cat$

If $\epsilon \in L$ we also allow a rule S→ϵ as long as S never appears on the RHS of a rule

Generalising from CNF

Add reduction rules:

- $\dfrac{}{(i,i):A}$ (REDUCE0) if A→ϵ
- $\dfrac{(i,j):B \quad (j,k):C \quad (k,l):D}{(i,l)\ :\ A}$(REDUCE3) if A→B C D
- $\dfrac{(i,j):B \quad (j,k):C \quad (k,l):D \quad (l,m):E}{(i,m)\ :\ A}$(REDUCE4) if A→B C D E
- ...

I'm not aware of any logic that would allow an infinite number of deduction rules, but for a given grammar you can cap it at the longest RHS that you have.
CKY parsing with these additional rules is less efficient.

Generalising from CNF

Add reduction rules:

- $\dfrac{}{(i,i) : A}$ (REDUCE0) if A→ϵ
- $\dfrac{(i,j) : B \quad (j,k) : C \quad (k,l) : D}{(i,l) \;:\; A}$ (REDUCE3) if A→B C D
- $\dfrac{(i,j) : B \quad (j,k) : C \quad (k,l) : D \quad (l,m) : E}{(i,m) \;:\; A}$ (REDUCE4) if A→B C D E
- ...

I'm not aware of any logic that would allow an infinite number of deduction rules, but for a given grammar you can cap it at the longest RHS that you have.

CKY parsing with these additional rules is less efficient.

Generalising from CNF

Add reduction rules:

- $\dfrac{}{(i,i):A}$ (REDUCE0) if A→ϵ
- $\dfrac{(i,j):B \quad (j,k):C \quad (k,l):D}{(i,l)\ :\ A}$ (REDUCE3) if A→B C D
- $\dfrac{(i,j):B \quad (j,k):C \quad (k,l):D \quad (l,m):E}{(i,m)\ :\ A}$ (REDUCE4) if A→B C D E
- ...

I'm not aware of any logic that would allow an infinite number of deduction rules, but for a given grammar you can cap it at the longest RHS that you have.

CKY parsing with these additional rules is less efficient.

Generalising from CNF

Add reduction rules:

- $\dfrac{}{(i,i):A}$ (REDUCE0) if A→ϵ
- $\dfrac{(i,j):B \quad (j,k):C \quad (k,l):D}{(i,l) \ : \ A}$ (REDUCE3) if A→B C D
- $\dfrac{(i,j):B \quad (j,k):C \quad (k,l):D \quad (l,m):E}{(i,m) \ : \ A}$ (REDUCE4) if A→B C D E
- ...

I'm not aware of any logic that would allow an infinite number of deduction rules, but for a given grammar you can cap it at the longest RHS that you have.

CKY parsing with these additional rules is less efficient.

Generalising from CNF

Add reduction rules:

- $\dfrac{}{(i,i) : A}$ (REDUCE0) if A$\rightarrow\epsilon$
- $\dfrac{(i,j) : B \quad (j,k) : C \quad (k,l) : D}{(i,l) \ : \ A}$ (REDUCE3) if A$\rightarrow$B C D
- $\dfrac{(i,j) : B \quad (j,k) : C \quad (k,l) : D \quad (l,m) : E}{(i,m) \ : \ A}$ (REDUCE4) if A$\rightarrow$B C D E
- ...

I'm not aware of any logic that would allow an infinite number of deduction rules, but for a given grammar you can cap it at the longest RHS that you have.

CKY parsing with these additional rules is less efficient.

Generalising from CNF

Add reduction rules:

- $\dfrac{}{(i,i):A}$ $(\textsc{reduce0})$ if A$\rightarrow\epsilon$
- $\dfrac{(i,j):B \quad (j,k):C \quad (k,l):D}{(i,l)\ :\ A}$ $(\textsc{reduce3})$ if A$\rightarrow$B C D
- $\dfrac{(i,j):B \quad (j,k):C \quad (k,l):D \quad (l,m):E}{(i,m)\ :\ A}$ $(\textsc{reduce4})$ if A$\rightarrow$B C D E
- ...

I'm not aware of any logic that would allow an infinite number of deduction rules, but for a given grammar you can cap it at the longest RHS that you have.

CKY parsing with these additional rules is less efficient.

Generalising from CNF

Add reduction rules:

- $\dfrac{}{(i,i):A}$ (REDUCE0) if A→ϵ
- $\dfrac{(i,j):B \quad (j,k):C \quad (k,l):D}{(i,l)\ :\ A}$ (REDUCE3) if A→B C D
- $\dfrac{(i,j):B \quad (j,k):C \quad (k,l):D \quad (l,m):E}{(i,m)\ :\ A}$ (REDUCE4) if A→B C D E
- ...

I'm not aware of any logic that would allow an infinite number of deduction rules, but for a given grammar you can cap it at the longest RHS that you have.
CKY parsing with these additional rules is less efficient.

CKY parsing

- Leave record of how each cell was filled (*backpointers*)
- Go back through the tree and use backpointers to extract derivation(s)

Backpointer: (RHS of rule used, partition of interval)

	j
i	LHS, $\{$(RHS$_1$,partition$_1$), (RHS$_2$,partition$_2$ $\ldots\}$

CKY parsing

- Leave record of how each cell was filled (*backpointers*)
- Go back through the tree and use backpointers to extract derivation(s)

Backpointer: (RHS of rule used, partition of interval)

	j
i	LHS, {(RHS_1,partition_1), (RHS_2,partition_2 ... }

CKY parsing

- S→DP VP
- DP→D N
- D→the, every
- N→cat, dog
- VP→slept, V DP
- V→saw

(6) $_0$ the $_1$ cat $_2$ slept $_3$

	1	2	3
0	D (the,(0,1))	DP (D N,(0,1),(1,2))	S (DP VP,(0,2),(2,3))
1		N (cat,(1,2))	
2			VP (slept,(2,3))

CKY parsing

- S→DP VP
- DP→D N
- D→the, every
- N→cat, dog
- VP→slept, V DP
- V→saw

(6) $_0$ the $_1$ cat $_2$ slept $_3$

	1	2	3
0	D (the,(0,1))	DP (D N,(0,1),(1,2))	S (DP VP,(0,2),(2,3))
1		N (cat,(1,2))	
2			VP (slept,(2,3))

CKY parsing

- S→DP VP
- DP→D N
- D→the, every
- N→cat, dog
- VP→slept, V DP
- V→saw

(6) $_0$ the $_1$ cat $_2$ slept $_3$

	1	2	3
0	D (the,(0,1))	DP (D N,(0,1),(1,2))	S (DP VP,(0,2),(2,3))
1		N (cat,(1,2))	
2			VP (slept,(2,3))

CKY parsing

- S→DP VP
- DP→D N
- D→the, every
- N→cat, dog
- VP→slept, V DP
- V→saw

(6) $_0$ the $_1$ cat $_2$ slept $_3$

	1	2	3
0	D (the,(0,1))	DP (D N,(0,1),(1,2))	S (DP VP,(0,2),(2,3))
1		N (cat,(1,2))	
2			VP (slept,(2,3))

CKY parsing

- S→DP VP
- DP→D N
- D→the, every
- N→cat, dog
- VP→slept, V DP
- V→saw

(6) $_0$ the $_1$ cat $_2$ slept $_3$

	1	2	3
0	D (the,(0,1))	DP (D N,(0,1),(1,2))	S (DP VP,(0,2),(2,3))
1		N (cat,(1,2))	
2			VP (slept,(2,3))

CKY parsing

	1	2	3
0	D (the,0)	DP (D N,1)	S (DP VP,2)
1		N (cat,0)	
2			VP (slept,0)

$_0$ the $_1$ cat $_2$ slept $_3$

DP: (0,2) with partition 1: D in (0,1), N in (1,2)

S: (0,3) with partition 2: DP in (0,2), VP in (2,3)

cell (i,j) with RHS w and partition 0: no need to look

cell (i,j) with RHS A B and partition k: look in (i,i+k) for A; look in (i+k,j) for B

CKY parsing

	1	2	3
0	D (the,0)	DP (D N,1)	S (DP VP,2)
1		N (cat,0)	
2			VP (slept,0)

$_0$ the $_1$ cat $_2$ slept $_3$

DP: (0,2) with partition 1: D in (0,1), N in (1,2)

S: (0,3) with partition 2: DP in (0,2), VP in (2,3)

cell (i,j) with RHS w and partition 0: no need to look

cell (i,j) with RHS A B and partition k: look in (i,i+k) for A; look in (i+k,j) for B

CKY parsing

	1	2	3
0	D (the,0)	DP (D N,1)	S (DP VP,2)
1		N (cat,0)	
2			VP (slept,0)

$_0$ the $_1$ cat $_2$ slept $_3$

DP: (0,2) with partition 1: D in (0,1), N in (1,2)

S: (0,3) with partition 2: DP in (0,2), VP in (2,3)

cell (i,j) with RHS w and partition 0: no need to look

cell (i,j) with RHS A B and partition k: look in (i,i+k) for A; look in (i+k,j) for B

CKY parsing

	1	2	3
0	D (the,0)	DP (D N,1)	S (DP VP,2)
1		N (cat,0)	
2			VP (slept,0)

$_0$ the $_1$ cat $_2$ slept $_3$

DP: (0,2) with partition 1: D in (0,1), N in (1,2)

S: (0,3) with partition 2: DP in (0,2), VP in (2,3)

cell (i,j) with RHS w and partition 0: no need to look

cell (i,j) with RHS A B and partition k: look in (i,i+k) for A; look in (i+k,j) for B

CKY parsing

	1	2	3
0	D (the,0)	DP (D N,1)	S (DP VP,2)
1		N (cat,0)	
2			VP (slept,0)

$_0$ the $_1$ cat $_2$ slept $_3$

DP: (0,2) with partition 1: D in (0,1), N in (1,2)

S: (0,3) with partition 2: DP in (0,2), VP in (2,3)

cell (i,j) with RHS w and partition 0: no need to look

cell (i,j) with RHS A B and partition k: look in (i,i+k) for A; look in (i+k,j) for B

CKY parsing

	1	2	3
0	D (the,0)	DP (D N,1)	S (DP VP,2)
1		N (cat,0)	
2			VP (slept,0)

$_0$ the $_1$ cat $_2$ slept $_3$

DP: (0,2) with partition 1: D in (0,1), N in (1,2)

S: (0,3) with partition 2: DP in (0,2), VP in (2,3)

cell (i,j) with RHS w and partition 0: no need to look

cell (i,j) with RHS A B and partition k: look in (i,i+k) for A; look in (i+k,j) for B

CKY parsing

	1	2	3
0	D (the,0)	DP (D N,1)	S (DP VP,2)
1		N (cat,0)	
2			VP (slept,0)

$_0$ the $_1$ cat $_2$ slept $_3$

DP: (0,2) with partition 1: D in (0,1), N in (1,2)

S: (0,3) with partition 2: DP in (0,2), VP in (2,3)

cell (i,j) with RHS w and partition 0: no need to look

cell (i,j) with RHS A B and partition k: look in (i,i+k) for A; look in (i+k,j) for B

CKY parsing

	1	2	3
0	D (the,0)	DP (D N,1)	S (DP VP,2)
1		N (cat,0)	
2			VP (slept,0)

$_0$ the $_1$ cat $_2$ slept $_3$

DP: (0,2) with partition 1: D in (0,1), N in (1,2)

S: (0,3) with partition 2: DP in (0,2), VP in (2,3)

cell (i,j) with RHS w and partition 0: no need to look

cell (i,j) with RHS A B and partition k: look in (i,i+k) for A; look in (i+k,j) for B

CKY parsing: practice

Add backpointers to your charts:

- S→DP VP
- DP→D NP
- D→the, a
- NP→man, woman, cat, telescope, NP PP
- PP→P DP
- P→with, on, to
- VP→slept, fell, V DP, VP PP
- V→saw, hit

(7) The woman with the cat fell

(8) The man saw the woman with the telescope

(9) The woman the cat with

CKY parsing: practice

Add backpointers to your charts:

- S→DP VP
- DP→D NP
- D→the, a
- NP→man, woman, cat, telescope, NP PP
- PP→P DP
- P→with, on, to
- VP→slept, fell, V DP, VP PP
- V→saw, hit

(7) The woman with the cat fell

(8) The man saw the woman with the telescope

(9) The woman the cat with

CKY parsing: practice

Add backpointers to your charts:

- S→DP VP
- DP→D NP
- D→the, a
- NP→man, woman, cat, telescope, NP PP
- PP→P DP
- P→with, on, to
- VP→slept, fell, V DP, VP PP
- V→saw, hit

(7) The woman with the cat fell

(8) The man saw the woman with the telescope

(9) The woman the cat with

CKY parsing: practice

Add backpointers to your charts:

- S→DP VP
- DP→D NP
- D→the, a
- NP→man, woman, cat, telescope, NP PP
- PP→P DP
- P→with, on, to
- VP→slept, fell, V DP, VP PP
- V→saw, hit

(7) The woman with the cat fell

(8) The man saw the woman with the telescope

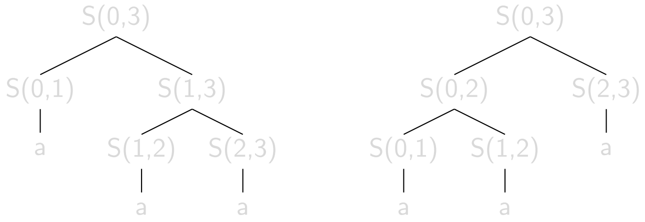
(9) The woman the cat with

Tree collector

G_{a^+}

$$\begin{array}{lcl} S & \to & S\ S \\ S & \to & a \end{array}$$

	1	2	3
0	S {(a,0)}	S {(SS,1)}	S {(SS,1),(SS,2)}
1		S {(a,0)}	S {(SS,1)}
2			S {(a,0)}

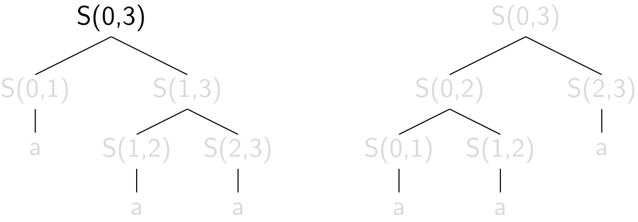


Tree collector

G_{a^+}

S	$\rightarrow$	S S
S	$\rightarrow$	a

	1	2	3
0	S {(a,0)}	S {(SS,1)}	S {(SS,1),(SS,2)}
1		S {(a,0)}	S {(SS,1)}
2			S {(a,0)}

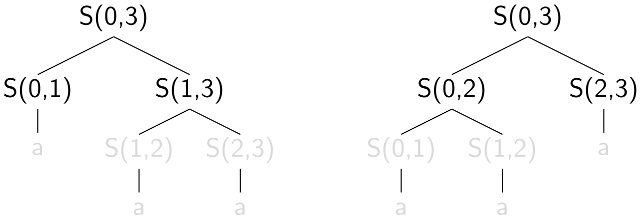


Tree collector

G_{a^+}

$$
\begin{array}{lcl}
S & \rightarrow & S\ S \\
S & \rightarrow & a
\end{array}
$$

	1	2	3
0	S {(a,0)}	S {(SS,1)}	S {(SS,1),(SS,2)}
1		S {(a,0)}	S {(SS,1)}
2			S {(a,0)}

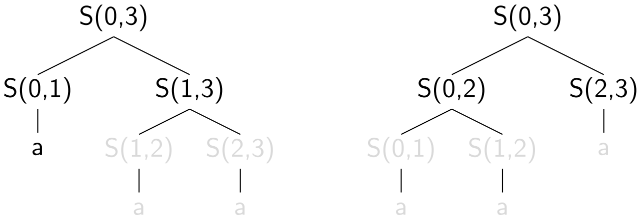


Tree collector

G_{a^+}

$$\begin{array}{lcl} S & \rightarrow & S\ S \\ S & \rightarrow & a \end{array}$$

	1	2	3
0	S {(a,0)}	S {(SS,1)}	S {(SS,1),(SS,2)}
1		S {(a,0)}	S {(SS,1)}
2			S {(a,0)}

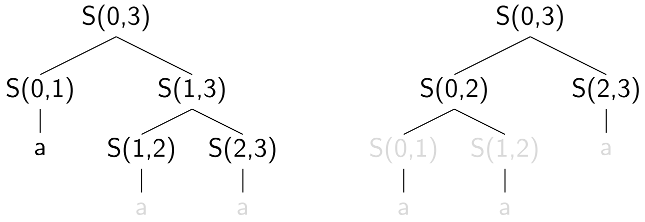


Tree collector

G_{a^+}

$$
\begin{array}{lcl}
S & \rightarrow & S\ S \\
S & \rightarrow & a
\end{array}
$$

	1	2	3
0	S {(a,0)}	S {(SS,1)}	S {(SS,1),(SS,2)}
1		S {(a,0)}	S {(SS,1)}
2			S {(a,0)}

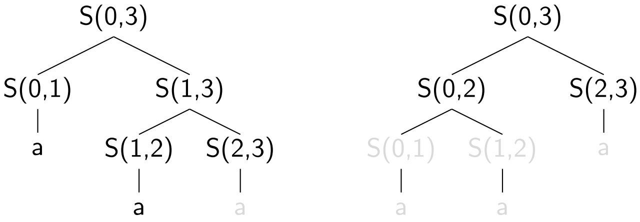


Tree collector

G_{a^+}

$$
\begin{array}{lcl}
S & \rightarrow & S\ S \\
S & \rightarrow & a
\end{array}
$$

	1	2	3
0	S {(a,0)}	S {(SS,1)}	S {(SS,1),(SS,2)}
1		S {(a,0)}	S {(SS,1)}
2			S {(a,0)}

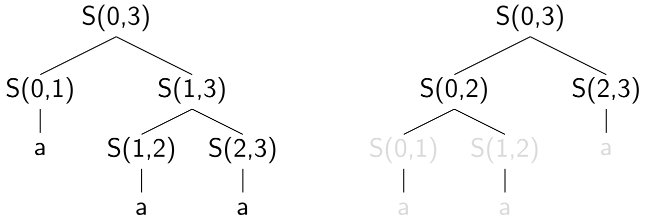


Tree collector

G_{a^+}

$$\begin{array}{lcl} S & \rightarrow & S\ S \\ S & \rightarrow & a \end{array}$$

	1	2	3
0	S {(a,0)}	S {(SS,1)}	S {(SS,1),(SS,2)}
1		S {(a,0)}	S {(SS,1)}
2			S {(a,0)}

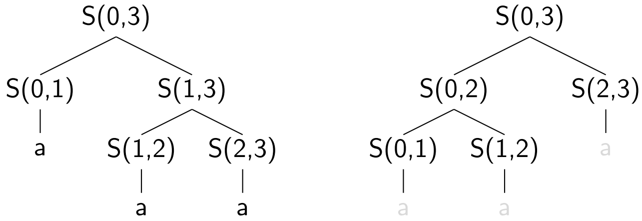


Tree collector

G_{a^+}

$$
\begin{array}{lcl}
S & \rightarrow & S\ S \\
S & \rightarrow & a
\end{array}
$$

	1	2	3
0	S {(a,0)}	S {(SS,1)}	S {(SS,1),(SS,2)}
1		S {(a,0)}	S {(SS,1)}
2			S {(a,0)}

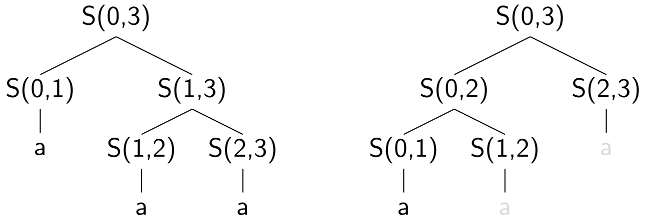


Tree collector

G_{a^+}

$$\begin{array}{lll} S & \rightarrow & S\ S \\ S & \rightarrow & a \end{array}$$

	1	2	3
0	S {(a,0)}	S {(SS,1)}	S {(SS,1),(SS,2)}
1		S {(a,0)}	S {(SS,1)}
2			S {(a,0)}

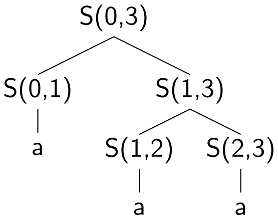


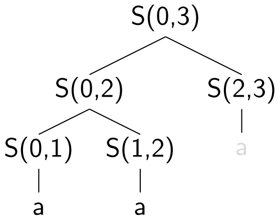
Tree collector

G_{a^+}

$$
\begin{array}{lll}
S & \rightarrow & S\ S \\
S & \rightarrow & a
\end{array}
$$

	1	2	3
0	S {(a,0)}	S {(SS,1)}	S {(SS,1),(SS,2)}
1		S {(a,0)}	S {(SS,1)}
2			S {(a,0)}



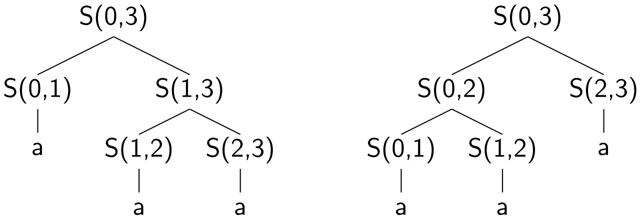


Tree collector

G_{a^+}

$$
\begin{array}{lcl}
S & \rightarrow & S\ S \\
S & \rightarrow & a
\end{array}
$$

	1	2	3
0	S {(a,0)}	S {(SS,1)}	S {(SS,1),(SS,2)}
1		S {(a,0)}	S {(SS,1)}
2			S {(a,0)}



Tree collector

1. For each start category S in cell (0,n), start a tree with the S(0,n) at the root. Start at the root of the first tree.
2. If you're at node N(i,j): Look in cell (i,j), category N. For each backpointer (A B, k), a copy of the tree so far should expand N(i,j) to A(i,i+k) and B(i+k,j). For each backpointer $(a, 0)$, a copy of the tree so far should expand N(i,j) to a
3. Traverse the tree in preorder fashion until you get to a nonterminal leaf. If you run out of tree, go to the root of the next tree. If you run out of trees, you're done. Go to the last step.
4. Go back to step 2
5. For each tree, delete the (i,j) indices. Now you have all parses of your string with your grammar.

Exercise (Tree collecting)

1. *Add backpointers to your charts if necessary and extract trees.*
2. *If you haven't yet, parse aaaa with G_{a^+} and extract the trees*

Tree collector

1. For each start category S in cell (0,n), start a tree with the S(0,n) at the root. Start at the root of the first tree.
2. If you're at node N(i,j): Look in cell (i,j), category N. For each backpointer (A B, k), a copy of the tree so far should expand N(i,j) to A(i,i+k) and B(i+k,j). For each backpointer $(a, 0)$, a copy of the tree so far should expand N(i,j) to a
3. Traverse the tree in preorder fashion until you get to a nonterminal leaf. If you run out of tree, go to the root of the next tree. If you run out of trees, you're done. Go to the last step.
4. Go back to step 2
5. For each tree, delete the (i,j) indices. Now you have all parses of your string with your grammar.

Exercise (Tree collecting)

1. *Add backpointers to your charts if necessary and extract trees.*
2. *If you haven't yet, parse aaaa with G_{a^+} and extract the trees*

Tree collector

1. For each start category S in cell (0,n), start a tree with the S(0,n) at the root. Start at the root of the first tree.
2. If you're at node N(i,j): Look in cell (i,j), category N. For each backpointer (A B, k), a copy of the tree so far should expand N(i,j) to A(i,i+k) and B(i+k,j). For each backpointer $(a, 0)$, a copy of the tree so far should expand N(i,j) to a
3. Traverse the tree in preorder fashion until you get to a nonterminal leaf. If you run out of tree, go to the root of the next tree. If you run out of trees, you're done. Go to the last step.
4. Go back to step 2
5. For each tree, delete the (i,j) indices. Now you have all parses of your string with your grammar.

Exercise (Tree collecting)

1. *Add backpointers to your charts if necessary and extract trees.*
2. *If you haven't yet, parse aaaa with G_{a^+} and extract the trees*

Tree collector

1. For each start category S in cell (0,n), start a tree with the S(0,n) at the root. Start at the root of the first tree.
2. If you're at node N(i,j): Look in cell (i,j), category N. For each backpointer (A B, k), a copy of the tree so far should expand N(i,j) to A(i,i+k) and B(i+k,j). For each backpointer $(a, 0)$, a copy of the tree so far should expand N(i,j) to a
3. Traverse the tree in preorder fashion until you get to a nonterminal leaf. If you run out of tree, go to the root of the next tree. If you run out of trees, you're done. Go to the last step.
4. Go back to step 2
5. For each tree, delete the (i,j) indices. Now you have all parses of your string with your grammar.

Exercise (Tree collecting)

1. *Add backpointers to your charts if necessary and extract trees.*
2. *If you haven't yet, parse aaaa with G_{a^+} and extract the trees*

Tree collector

1. For each start category S in cell (0,n), start a tree with the S(0,n) at the root. Start at the root of the first tree.
2. If you're at node N(i,j): Look in cell (i,j), category N. For each backpointer (A B, k), a copy of the tree so far should expand N(i,j) to A(i,i+k) and B(i+k,j). For each backpointer $(a, 0)$, a copy of the tree so far should expand N(i,j) to a
3. Traverse the tree in preorder fashion until you get to a nonterminal leaf. If you run out of tree, go to the root of the next tree. If you run out of trees, you're done. Go to the last step.
4. Go back to step 2
5. For each tree, delete the (i,j) indices. Now you have all parses of your string with your grammar.

Exercise (Tree collecting)

1. *Add backpointers to your charts if necessary and extract trees.*
2. *If you haven't yet, parse aaaa with G_{a^+} and extract the trees*

Tree collector

1. For each start category S in cell (0,n), start a tree with the S(0,n) at the root. Start at the root of the first tree.
2. If you're at node N(i,j): Look in cell (i,j), category N. For each backpointer (A B, k), a copy of the tree so far should expand N(i,j) to A(i,i+k) and B(i+k,j). For each backpointer $(a, 0)$, a copy of the tree so far should expand N(i,j) to a
3. Traverse the tree in preorder fashion until you get to a nonterminal leaf. If you run out of tree, go to the root of the next tree. If you run out of trees, you're done. Go to the last step.
4. Go back to step 2
5. For each tree, delete the (i,j) indices. Now you have all parses of your string with your grammar.

Exercise (Tree collecting)

1. *Add backpointers to your charts if necessary and extract trees.*
2. *If you haven't yet, parse aaaa with G_{a^+} and extract the trees*

Tree collector

1. For each start category S in cell (0,n), start a tree with the S(0,n) at the root. Start at the root of the first tree.
2. If you're at node N(i,j): Look in cell (i,j), category N. For each backpointer (A B, k), a copy of the tree so far should expand N(i,j) to A(i,i+k) and B(i+k,j). For each backpointer $(a, 0)$, a copy of the tree so far should expand N(i,j) to a
3. Traverse the tree in preorder fashion until you get to a nonterminal leaf. If you run out of tree, go to the root of the next tree. If you run out of trees, you're done. Go to the last step.
4. Go back to step 2
5. For each tree, delete the (i,j) indices. Now you have all parses of your string with your grammar.

Exercise (Tree collecting)

1. *Add backpointers to your charts if necessary and extract trees.*
2. *If you haven't yet, parse aaaa with G_{a^+} and extract the trees*

References

Aho, Alfred V, and Jeffrey D Ullman. 1972. *The theory of parsing, translation, and compiling*. Prentice-Hall, Inc.

Shieber, Stuart M, Yves Schabes, and Fernando CN Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of logic programming* 24:3–36.