CS 784: Computational Linguistics
Lecture 13: Syntax: Constituency Parsing

Freda Shi

School of Computer Science, University of Waterloo
fhs@uwaterloo.ca

March 5, 2025

What is Syntax?

The cat near the children ____.

meow

meows

What is Syntax?

The cat near the children ____.

meow

meows ✓

- Rules, principles, and processes that govern the sentence structure of a language.

What is Syntax?

The cat near the children ____.

meow

meows ✓

- Rules, principles, and processes that govern the sentence structure of a language.
- Can differ widely among languages.

What is Syntax?

The cat near the children ____.

meow

meows ✓

- Rules, principles, and processes that govern the sentence structure of a language.
- Can differ widely among languages.
- Every language has some systematic structural principles.

What is Syntax?

The cat near the children ____.

meow

meows ✓

- Rules, principles, and processes that govern the sentence structure of a language.
- Can differ widely among languages.
- Every language has some systematic structural principles.

We use **grammar** to denote a formal object that represents the rules/principles/processes that determine sentence structure.

Subject, Verb, Object

Syntax determines the ordering of these components of a sentence.

Word order	English equivalent	Proportion of languages		Example languages
SOV	"Cows grass eat."	45%		Ancient Greek, Bengali, Burmese, Hindi/Urdu, Japanese, Korean, Latin, Persian, Sanskrit, Tamil, Telugu, Turkish, etc
SVO	"Cows eat grass."	42%		Chinese, English, French, Hausa, Hebrew, Arabic, Italian, Malay, Portuguese, Spanish, Swahili, Thai, Vietnamese, etc
VSO	"Eat cows grass."	9%		Biblical Hebrew, Classical Arabic, Filipino, Ge'ez, Irish, Māori, Tuareg-Berber, Welsh
VOS	"Eat grass cows."	3%		Car, Fijian, Malagasy, Q'eqchi', Terêna
OVS	"Grass eat cows."	1%		Hixkaryana, Urarina
OSV	"Grass cows eat."	0%		Tobati, Warao

Frequency distribution of word order in languages surveyed by Russell S. Tomlin in the 1980s[1][2] (v·t·e)

Subject, Verb, Object

Syntax determines the ordering of these components of a sentence.

Word order	English equivalent	Proportion of languages	Example languages
SOV	"Cows grass eat."	45%	Ancient Greek, Bengali, Burmese, Hindi/Urdu, Japanese, Korean, Latin, Persian, Sanskrit, Tamil, Telugu, Turkish, etc
SVO	"Cows eat grass."	42%	Chinese, English, French, Hausa, Hebrew, Arabic, Italian, Malay, Portuguese, Spanish, Swahili, Thai, Vietnamese, etc
VSO	"Eat cows grass."	9%	Biblical Hebrew, Classical Arabic, Filipino, Ge'ez, Irish, Māori, Tuareg-Berber, Welsh
VOS	"Eat grass cows."	3%	Car, Fijian, Malagasy, Q'eqchi', Terêna
OVS	"Grass eat cows."	1%	Hixkaryana, Urarina
OSV	"Grass cows eat."	0%	Tobati, Warao

Frequency distribution of word order in languages surveyed by Russell S. Tomlin in the 1980s[1][2] (v · t · e)

Hanh and Xu (PNAS 2022): Information-theoretic justification on word orders.

Phrase Structures/Constituency Grammar

Focuses on **constituent relations**.

Informal understanding: sentences have hierarchical structure.

Phrase Structures/Constituency Grammar

Focuses on **constituent relations**.

Informal understanding: sentences have hierarchical structure.

A sentence is made up of two pieces:

- Subject, typically a noun phrase (NP).
- Predicate, typically a verb phrase (VP).

Phrase Structures/Constituency Grammar

Focuses on **constituent relations**.

Informal understanding: sentences have hierarchical structure.

A sentence is made up of two pieces:

- Subject, typically a noun phrase (NP).
- Predicate, typically a verb phrase (VP).

NPs and VPs are made up of smaller pieces:

- `a cat = (a + cat)`
- `walked to the park = (walk + (to + (the + park) ) )`

Each parenthesized phrase is a constituent in the constituent parse.

Constituency Test

Constituent: a group of words that functions as a single unit.

Constituency Test

Constituent: a group of words that functions as a single unit.

How to formally determine constituents?

Constituency Test

Constituent: a group of words that functions as a single unit.

How to formally determine constituents?

Linguists (try to) determine constituents via constituency tests.

Constituency Test

Constituent: a group of words that functions as a single unit.

How to formally determine constituents?

Linguists (try to) determine constituents via constituency tests.

- A constituency test follows some rules to construct a new sentence, focusing on the constituent candidate of interests.

Constituency Test

Constituent: a group of words that functions as a single unit.

How to formally determine constituents?

Linguists (try to) determine constituents via constituency tests.

- A constituency test follows some rules to construct a new sentence, focusing on the constituent candidate of interests.
- If the constructed sentence looks good (to native speakers), we find some positive evidence about constituency.

Constituency Test

Constituent: a group of words that functions as a single unit.

How to formally determine constituents?

Linguists (try to) determine constituents via constituency tests.

- A constituency test follows some rules to construct a new sentence, focusing on the constituent candidate of interests.
- If the constructed sentence looks good (to native speakers), we find some positive evidence about constituency.

```
Drunks could put off the customers.
```

What are the possible constituents and why?

Coordination Test

Coordinate the candidate constituent with something else.

`Drunks could put off the customers.`

Coordination Test

Coordinate the candidate constituent with something else.

`Drunks could put off the customers.`

- `Drunks could [put off the customers] and sing.`

Coordination Test

Coordinate the candidate constituent with something else.

`Drunks could put off the customers.`

- `Drunks could [put off the customers] and sing.`
- `Drunks could put off [the customers] and their neighbors.`

Coordination Test

Coordinate the candidate constituent with something else.

`Drunks could put off the customers.`

- `Drunks could [put off the customers] and sing.`
- `Drunks could put off [the customers] and their neighbors.`
- `Drunks [could] and [would] put off the customers.`

Coordination Test

Coordinate the candidate constituent with something else.

`Drunks could put off the customers.`

- `Drunks could [put off the customers] and sing.`
- `Drunks could put off [the customers] and their neighbors.`
- `Drunks [could] and [would] put off the customers.`
- `* Drunks could and would put [off the] and […] customers.`

Coordination Test

Coordinate the candidate constituent with something else.

`Drunks could put off the customers.`

- `Drunks could [put off the customers] and sing.`
- `Drunks could put off [the customers] and their neighbors.`
- `Drunks [could] and [would] put off the customers.`
- `* Drunks could and would put [off the] and […] customers.`

Caveat: constituency tests are positive evidences but not necessary conditions.

Topicalization Test [1]

Move the candidate constituent to the front.

Modal adverbs can be added to improve naturalness.

```
Drunks could put off the customers.
```

[1] Topicalization is a mechanism of syntax that establishes an expression as the sentence or clause topic by having it appear at the front of the sentence or clause (as opposed to in a canonical position later in the sentence).

Topicalization Test [1]

Move the candidate constituent to the front.

Modal adverbs can be added to improve naturalness.

`Drunks could put off the customers.`

- …and `[the customers], drunks certainly could put off.`

[1] Topicalization is a mechanism of syntax that establishes an expression as the sentence or clause topic by having it appear at the front of the sentence or clause (as opposed to in a canonical position later in the sentence).

Topicalization Test [1]

Move the candidate constituent to the front.

Modal adverbs can be added to improve naturalness.

`Drunks could put off the customers.`

- `…and [the customers], drunks certainly could put off.`
- `* …and [customers], drunks could certainly put off the.`

[1] Topicalization is a mechanism of syntax that establishes an expression as the sentence or clause topic by having it appear at the front of the sentence or clause (as opposed to in a canonical position later in the sentence).

Deletion Test

Delete the span of interest.

Word orders can be changed to improve naturalness.

```
Drunks could put off the customers in the bar.
```

Deletion Test

Delete the span of interest.

Word orders can be changed to improve naturalness.

`Drunks could put off the customers in the bar.`

- `Drunks could put off the customers [`~~`in the bar`~~`].`

Deletion Test

Delete the span of interest.

Word orders can be changed to improve naturalness.

`Drunks could put off the customers in the bar.`

- `Drunks could put off the customers [~~in the bar~~].`
- `*Drunks could put off the customers [~~in the~~] bar.`

Substitution Test

Substitute the candidate constituent with the appropriate proform (pronoun/proverb/etc.).

Word orders can be changed to improve naturalness.

`Drunks could put off the customers.`

Substitution Test

Substitute the candidate constituent with the appropriate proform (pronoun/proverb/etc.).

Word orders can be changed to improve naturalness.

`Drunks could put off the customers.`

- `Drunks could [do so = put off the customers].`

Substitution Test

Substitute the candidate constituent with the appropriate proform (pronoun/proverb/etc.).

Word orders can be changed to improve naturalness.

`Drunks could put off the customers.`

- `Drunks could [do so = put off the customers].`
- `Drunks could put [them = the customers] off.`

Substitution Test

Substitute the candidate constituent with the appropriate proform (pronoun/proverb/etc.).

Word orders can be changed to improve naturalness.

`Drunks could put off the customers.`

- `Drunks could [do so = put off the customers].`
- `Drunks could put [them = the customers] off.`
- `*Drunks could put [them = customers] off the.`

Substitution Test

Substitute the candidate constituent with the appropriate proform (pronoun/proverb/etc.).

Word orders can be changed to improve naturalness.

`Drunks could put off the customers.`

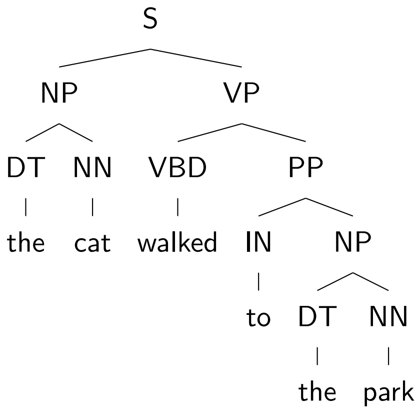
- `Drunks could [do so = put off the customers].`
- `Drunks could put [them = the customers] off.`
- `*Drunks could put [them = customers] off the.`

Being a constituent does not necessitate passing all tests.

But if a group of words is a constituent, it should pass at least one test.

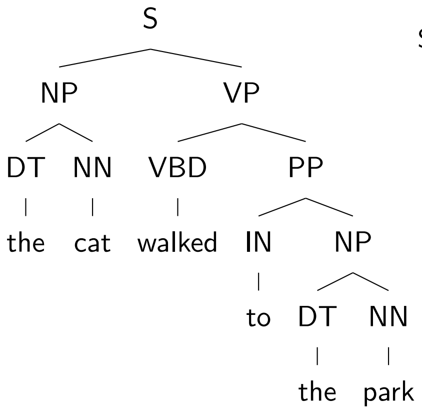
Constituency Parsing as Bracketing

Which spans are constituents in a sentence?



Constituency Parsing as Bracketing

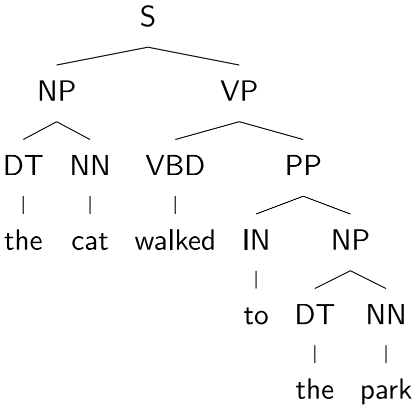
Which spans are constituents in a sentence?



$S \rightarrow \quad$ (S NP VP)

Constituency Parsing as Bracketing

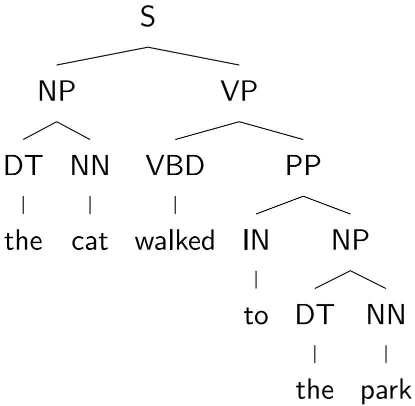
Which spans are constituents in a sentence?



$$
\begin{aligned}
\text{S} &\rightarrow \text{(S NP VP)} \\
&\rightarrow \text{(S (NP DT NN) VP)}
\end{aligned}
$$

Constituency Parsing as Bracketing

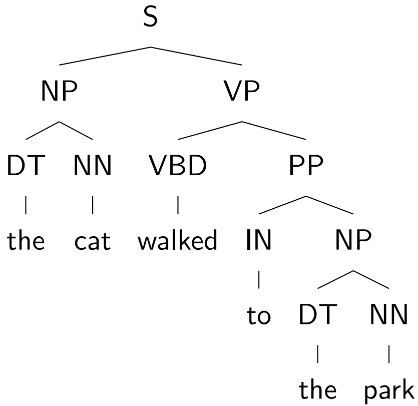
Which spans are constituents in a sentence?



$$
\begin{aligned}
\text{S} &\rightarrow \quad \text{(S NP VP)} \\
&\rightarrow \quad \text{(S (NP DT NN) VP)} \\
&\rightarrow \quad \ldots
\end{aligned}
$$

Constituency Parsing as Bracketing

Which spans are constituents in a sentence?

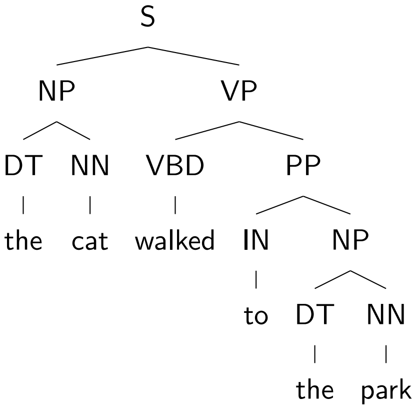


$$
\begin{aligned}
\text{S} \rightarrow\;& \text{(S NP VP)} \\
\rightarrow\;& \text{(S (NP DT NN) VP)} \\
\rightarrow\;& \ldots
\end{aligned}
$$

(S
 (NP (DT the) (NN cat))
 (VP (VBD walked)
  (PP (IN to)
   (NP (DT the) (NN park))
)))

Constituency Parsing as Bracketing

Which spans are constituents in a sentence?

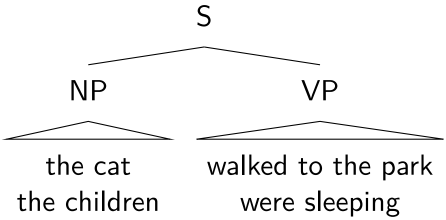


$$
\begin{aligned}
\text{S} \rightarrow\ & \text{(S NP VP)} \\
\rightarrow\ & \text{(S (NP DT NN) VP)} \\
\rightarrow\ & \dots
\end{aligned}
$$

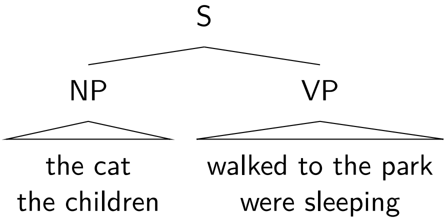
(S
 (NP (DT the) (NN cat))
 (VP (VBD walked)
  (PP (IN to)
   (NP (DT the) (NN park))
)))

The brackets and trees are mutually translatable.

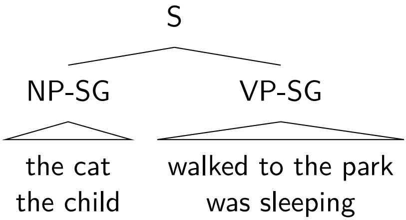
Labels as Syntactic Substitutability



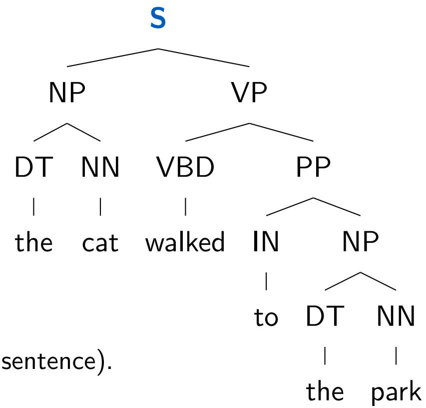
Labels as Syntactic Substitutability



Constraints (e.g., singular/plural labels) are necessary to ensure grammaticality.

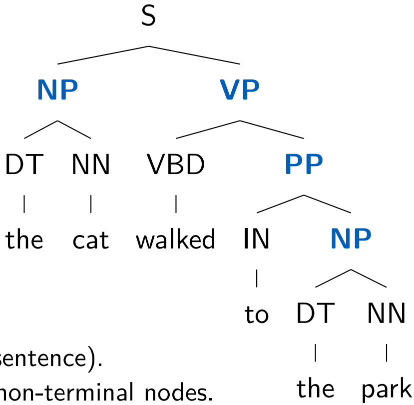


Types of Nodes



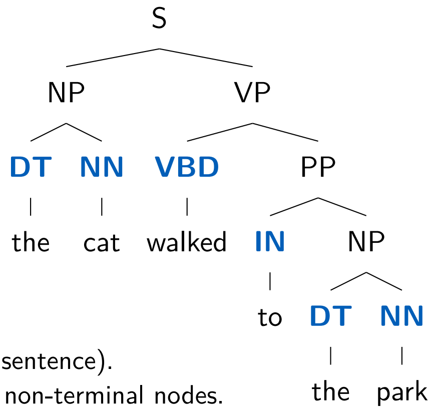
- **S**: root node (sentence).

Types of Nodes



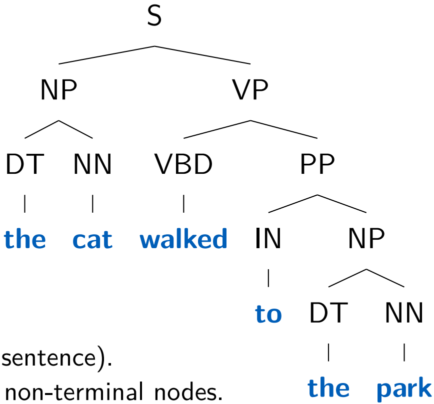
- **S**: root node (sentence).
- **NP, VP, PP**: non-terminal nodes.

Types of Nodes



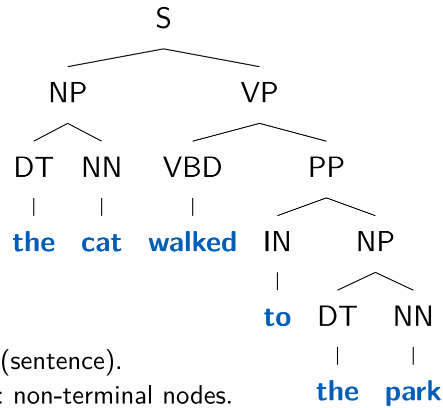
- **S**: root node (sentence).
- **NP, VP, PP**: non-terminal nodes.
- **DT, NN, VBD, IN**: pre-terminal nodes.

Types of Nodes



- **S**: root node (sentence).
- **NP, VP, PP**: non-terminal nodes.
- **DT, NN, VBD, IN**: pre-terminal nodes.
- **the, cat, walked, to, park**: terminal nodes.

Types of Nodes



- **S**: root node (sentence).
- **NP, VP, PP**: non-terminal nodes.
- **DT, NN, VBD, IN**: pre-terminal nodes.
- **the, cat, walked, to, park**: terminal nodes.

The Penn treebank tagset: `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`

Head of a Constituent

The **head** of a constituent is the most responsible/important word for the constituent label.

Head of a Constituent

The **head** of a constituent is the most responsible/important word for the constituent label.

`The cat walked to the park.`

Head of a Constituent

The **head** of a constituent is the most responsible/important word for the constituent label.

`The cat walked to the park.`

Q: Which word makes `the cat` a noun phrase (NP)?

A: .

Head of a Constituent

The **head** of a constituent is the most responsible/important word for the constituent label.

`The cat walked to the park.`

Q: Which word makes `the cat` a noun phrase (NP)?

A: `cat`.

Head of a Constituent

The **head** of a constituent is the most responsible/important word for the constituent label.

`The cat walked to the park.`

Q: Which word makes `the cat` a noun phrase (NP)?

A: `cat`.

Q: Which word makes `walked to the park` a verb phrase (VP)?

A: .

Head of a Constituent

The **head** of a constituent is the most responsible/important word for the constituent label.

`The cat walked to the park.`

Q: Which word makes `the cat` a noun phrase (NP)?

A: `cat`.

Q: Which word makes `walked to the park` a verb phrase (VP)?

A: `walked`.

Head of a Constituent

The **head** of a constituent is the most responsible/important word for the constituent label.

`The cat walked to the park.`

Q: Which word makes `the cat` a noun phrase (NP)?

A: `cat`.

Q: Which word makes `walked to the park` a verb phrase (VP)?

A: `walked`.

The concept of head is crucial to connect the constituency and dependency syntax.

Head of a Constituent

The **head** of a constituent is the most responsible/important word for the constituent label.

`The cat walked to the park.`

Q: Which word makes `the cat` a noun phrase (NP)?

A: `cat`.

Q: Which word makes `walked to the park` a verb phrase (VP)?

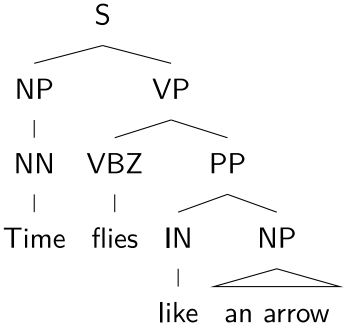
A: `walked`.

The concept of head is crucial to connect the constituency and dependency syntax.

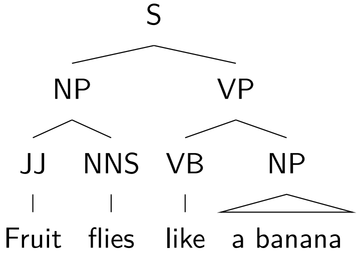
Caveat: There is room for ambiguity from the head concept above—in practice, Magerman (1995) and Collins (1999) propose head rules written by hand.

Syntactic Ambiguities

Time flies like an arrow.

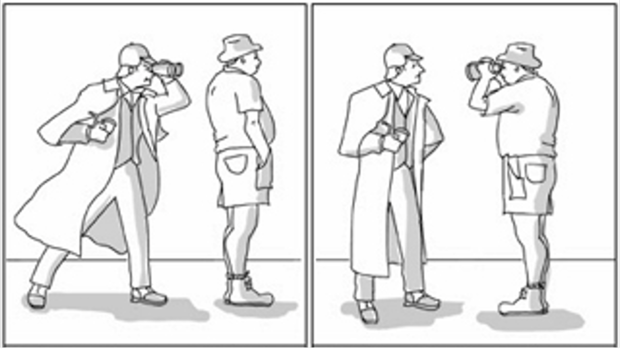


Fruit flies like a banana.



Syntactic Ambiguities: PP Attachment

Sherlock saw the man using binoculars.



Syntactic Ambiguities: Coordination Ambiguity

Consider the instruction below received by a robot:

```
Run and jump twice.
```

Syntactic Ambiguities: Coordination Ambiguity

Consider the instruction below received by a robot:

```
Run and jump twice.
```

Do you expect it to do

- Run, Jump, Jump, or
- Run, Jump, Run, Jump?

Syntactic Ambiguities: Coordination Ambiguity

Consider the instruction below received by a robot:

`Run and jump twice.`

Do you expect it to do

- Run, Jump, Jump, or
- Run, Jump, Run, Jump?

This coordination ambiguity has been a major issue of some neuro-symbolic models in language-based robot navigation (Mao et al., 2021).

Syntactic Ambiguities: Noun Compound and Adjective-Noun Ambiguity

`River boat race.`

Syntactic Ambiguities: Noun Compound and Adjective-Noun Ambiguity

`River boat race.`

Is that a boat race on a river or a “river-boat” race?

Syntactic Ambiguities: Noun Compound and Adjective-Noun Ambiguity

`River boat race.`

Is that a boat race on a river or a “river-boat” race?

`Ancient history teacher.`

Syntactic Ambiguities: Noun Compound and Adjective-Noun Ambiguity

`River boat race.`

Is that a boat race on a river or a “river-boat” race?

`Ancient history teacher.`

Is that a history teacher who teaches ancient history or a history teacher in the ancient era?

Syntactic Ambiguities: Noun Compound and Adjective-Noun Ambiguity

`River boat race.`

Is that a boat race on a river or a “river-boat” race?

`Ancient history teacher.`

Is that a history teacher who teaches ancient history or a history teacher in the ancient era?

Connecting to the real-world context is the key to (possibly) resolve the ambiguity.

Garden-Path Sentences

What is (should be) the next token?

`The horse raced past the barn`

Garden-Path Sentences

What is (should be) the next token?

`The horse raced past the barn .`

Garden-Path Sentences

What is (should be) the next token?

`The horse raced past the barn fell.`

Garden-Path Sentences

What is (should be) the next token?

`The horse raced past the barn fell.`

`The old man`

Garden-Path Sentences

What is (should be) the next token?

`The horse raced past the barn fell.`

`The old man the boat.`

Garden-Path Sentences

What is (should be) the next token?

`The horse raced past the barn fell.`

`The old man the boat.`

Garden-Path sentences: grammatically correct sentences that start in a way where readers' most likely interpretation will be incorrect.

Garden path: leading someone down/up the garden path.

Garden-Path Sentences

What is (should be) the next token?

`The horse raced past the barn fell.`

`The old man the boat.`

Garden-Path sentences: grammatically correct sentences that start in a way where readers' most likely interpretation will be incorrect.

Garden path: leading someone down/up the garden path.

Can be interpreted by explicitly drawing out the constituent parse tree.

Context-Free Grammars

A **grammar** defines a set of rules that define all possible strings in a language.

Context-Free Grammars

A **grammar** defines a set of rules that define all possible strings in a language.

A **context-free grammar** (CFG) is a tuple $(\mathcal{N}, \mathcal{T}, \mathcal{R}, S)$.

- $\mathcal{N}$: set of non-terminal symbols.
- $\mathcal{T}$: set of (pre-)terminal symbols.
- $\mathcal{R}$: set of rewriting rules.
- S: start symbol.

Context-Free Grammars

A **grammar** defines a set of rules that define all possible strings in a language.

A **context-free grammar** (CFG) is a tuple $(\mathcal{N}, \mathcal{T}, \mathcal{R}, S)$.

- $\mathcal{N}$: set of non-terminal symbols.
- $\mathcal{T}$: set of (pre-)terminal symbols.
- $\mathcal{R}$: set of rewriting rules.
- S: start symbol.

A rule is of the form $A \to \alpha$, where $A \in \mathcal{N}$ and $\alpha \in (\mathcal{N} \cup \mathcal{T})^*$.

Context-Free Grammars

A **grammar** defines a set of rules that define all possible strings in a language.

A **context-free grammar** (CFG) is a tuple $(\mathcal{N}, \mathcal{T}, \mathcal{R}, S)$.

- $\mathcal{N}$: set of non-terminal symbols.
- $\mathcal{T}$: set of (pre-)terminal symbols.
- $\mathcal{R}$: set of rewriting rules.
- S: start symbol.

A rule is of the form $A \rightarrow \alpha$, where $A \in \mathcal{N}$ and $\alpha \in (\mathcal{N} \cup \mathcal{T})^*$.

A parse tree is a derivation of a sentence from the start symbol S.

Context-Free Grammars

A **grammar** defines a set of rules that define all possible strings in a language.

A **context-free grammar** (CFG) is a tuple $(\mathcal{N}, \mathcal{T}, \mathcal{R}, S)$.

- $\mathcal{N}$: set of non-terminal symbols.
- $\mathcal{T}$: set of (pre-)terminal symbols.
- $\mathcal{R}$: set of rewriting rules.
- S: start symbol.

A rule is of the form $A \rightarrow \alpha$, where $A \in \mathcal{N}$ and $\alpha \in (\mathcal{N} \cup \mathcal{T})^*$.

A parse tree is a derivation of a sentence from the start symbol S.

Context-free: the application of a rule does not depend on the context.

An Example CFG

- $\mathcal{N} = \{S, NP, VP\}$
- $\mathcal{T} = \{\text{a}, \text{cat}, \text{meows}\}$
- $\mathcal{R} =$ the set containing the following rules:
  - $S \rightarrow NP\ VP$
  - $NP \rightarrow$ a cat
  - $VP \rightarrow$ meows
- $S = S$

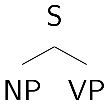
An Example CFG

- $\mathcal{N} = \{S, NP, VP\}$
- $\mathcal{T} = \{\text{a}, \text{cat}, \text{meows}\}$
- $\mathcal{R}$ = the set containing the following rules:
  - $S \rightarrow NP\ VP$
  - $NP \rightarrow$ a cat
  - $VP \rightarrow$ meows
- $S = S$

S

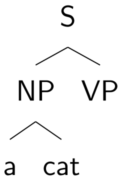
An Example CFG

- $\mathcal{N} = \{S, NP, VP\}$
- $\mathcal{T} = \{\text{a}, \text{cat}, \text{meows}\}$
- $\mathcal{R}$ = the set containing the following rules:
  - $S \rightarrow NP\ VP$
  - $NP \rightarrow$ a cat
  - $VP \rightarrow$ meows
- $S = S$



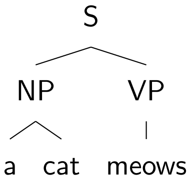
An Example CFG

- $\mathcal{N} = \{S, NP, VP\}$
- $\mathcal{T} = \{\text{a}, \text{cat}, \text{meows}\}$
- $\mathcal{R}$ = the set containing the following rules:
  - $S \rightarrow NP\ VP$
  - $NP \rightarrow$ a cat
  - $VP \rightarrow$ meows
- $S = S$



An Example CFG

- $\mathcal{N} = \{S, NP, VP\}$
- $\mathcal{T} = \{\text{a}, \text{cat}, \text{meows}\}$
- $\mathcal{R}$ = the set containing the following rules:
  - $S \rightarrow NP\ VP$
  - $NP \rightarrow$ a cat
  - $VP \rightarrow$ meows
- $S = S$



Weighted Context-Free Grammars

A **weighted context-free grammar** (WCFG) is a tuple $(\mathcal{N}, \mathcal{T}, \mathcal{R}, S, W)$.

- $\mathcal{N}$: set of non-terminal symbols.
- $\mathcal{T}$: set of (pre-)terminal symbols.
- $\mathcal{R}$: set of rules.
- S: start symbol.
- W: model parameters—weights for rules.

Weighted Context-Free Grammars

A **weighted context-free grammar** (WCFG) is a tuple $(\mathcal{N}, \mathcal{T}, \mathcal{R}, S, W)$.

- $\mathcal{N}$: set of non-terminal symbols.
- $\mathcal{T}$: set of (pre-)terminal symbols.
- $\mathcal{R}$: set of rules.
- S: start symbol.
- W: model parameters—weights for rules.

The score of a parse tree is the **product** of the weights of the rules used in the derivation.

Probabilistic Context-Free Grammars

A **probabilistic context-free grammar** (PCFG) is a WCFG where the weights are probabilities.

- $\mathcal{N}$: set of non-terminal symbols.
- $\mathcal{T}$: set of (pre-)terminal symbols.
- $\mathcal{R}$: set of rules.
- S: start symbol.
- P: model parameters—probabilities for rules.

Probabilistic Context-Free Grammars

A **probabilistic context-free grammar** (PCFG) is a WCFG where the weights are probabilities.

- $\mathcal{N}$: set of non-terminal symbols.
- $\mathcal{T}$: set of (pre-)terminal symbols.
- $\mathcal{R}$: set of rules.
- S: start symbol.
- P: model parameters—probabilities for rules.
  For each non-terminal A and its rules $\{A \to \alpha_1, A \to \alpha_2, \ldots\}$, we have

$$\sum_i P(A \to \alpha_i) = 1.$$

Probabilistic Context-Free Grammars

A **probabilistic context-free grammar** (PCFG) is a WCFG where the weights are probabilities.

- $\mathcal{N}$: set of non-terminal symbols.
- $\mathcal{T}$: set of (pre-)terminal symbols.
- $\mathcal{R}$: set of rules.
- S: start symbol.
- P: model parameters—probabilities for rules.
  For each non-terminal A and its rules $\{A \to \alpha_1, A \to \alpha_2, \ldots\}$, we have

$$\sum_i P(A \to \alpha_i) = 1.$$

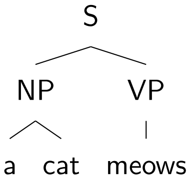
The probability of a parse tree is the **product** of the probabilities of the rules used in the derivation.

An Example PCFG

- $S \rightarrow NP\ VP$ [1.0]
- $NP \rightarrow$ a cat [0.4]
- $NP \rightarrow$ the cat [0.6]
- $VP \rightarrow$ meows [0.8]
- $VP \rightarrow$ sleeps [0.2]

An Example PCFG

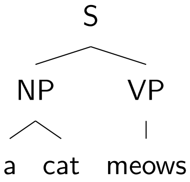
- $S \rightarrow NP\ VP$ [1.0]
- $NP \rightarrow$ a cat [0.4]
- $NP \rightarrow$ the cat [0.6]
- $VP \rightarrow$ meows [0.8]
- $VP \rightarrow$ sleeps [0.2]



Q: What is the probability of this parse tree?

An Example PCFG

- $S \rightarrow NP\ VP$ [1.0]
- $NP \rightarrow$ a cat [0.4]
- $NP \rightarrow$ the cat [0.6]
- $VP \rightarrow$ meows [0.8]
- $VP \rightarrow$ sleeps [0.2]

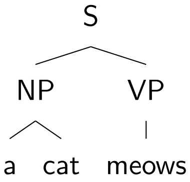


Q: What is the probability of this parse tree?

A: $1.0 \times 0.4 \times 0.8 = 0.32$.

An Example PCFG

- $S \rightarrow NP\ VP$ [1.0]
- $NP \rightarrow$ a cat [0.4]
- $NP \rightarrow$ the cat [0.6]
- $VP \rightarrow$ meows [0.8]
- $VP \rightarrow$ sleeps [0.2]



Q: What is the probability of this parse tree?

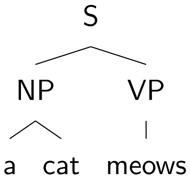
A: $1.0 \times 0.4 \times 0.8 = 0.32$.

(Related to Assignment 2)

Is it always the case that $\sum_{s \in \mathcal{L}} P(s) = 1$, where $\mathcal{L}$ is the language defined by the associated CFG? Why or why not?

An Example PCFG

- $S \rightarrow NP\ VP$ [1.0]
- $NP \rightarrow$ a cat [0.4]
- $NP \rightarrow$ the cat [0.6]
- $VP \rightarrow$ meows [0.8]
- $VP \rightarrow$ sleeps [0.2]



Q: What is the probability of this parse tree?

A: $1.0 \times 0.4 \times 0.8 = 0.32$.

(Related to Assignment 2)

Is it always the case that $\sum_{s \in \mathcal{L}} P(s) = 1$, where $\mathcal{L}$ is the language defined by the associated CFG? Why or why not?

Check out Hale (2003) for more!

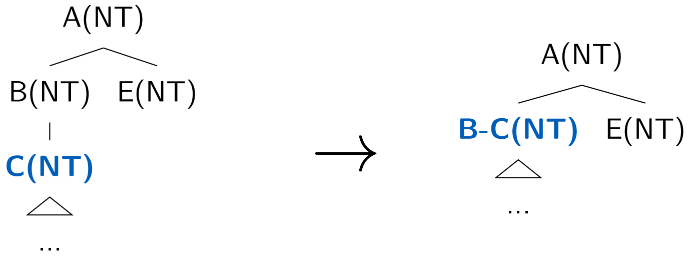
The Chomsky Normal Form: Unary Branches

For simplicity, let's assume we only work with **binary** constituency parse trees, where every non-terminal node has exactly two children nodes, i.e., the **Chomsky normal form**.

The Chomsky Normal Form: Unary Branches

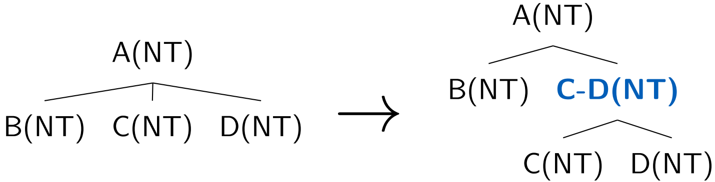
For simplicity, let's assume we only work with **binary** constituency parse trees, where every non-terminal node has exactly two children nodes, i.e., the **Chomsky normal form**.

A unary branch is collapsed into one node.



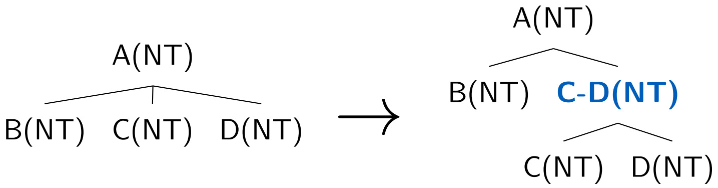
The Chomsky Normal Form: Ternary Branches

A ternary branch is split into two binary branches.



The Chomsky Normal Form: Ternary Branches

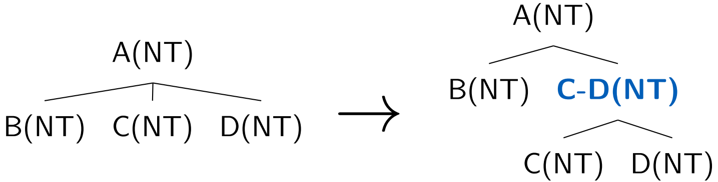
A ternary branch is split into two binary branches.



The split order is arbitrary: alternatively, we can split the branch from right to left.

The Chomsky Normal Form: Ternary Branches

A ternary branch is split into two binary branches.



The split order is arbitrary: alternatively, we can split the branch from right to left.

It's provable that any CFG can be converted to an equivalent CFG in Chomsky normal form.

Constituency Parsing as an NLP Task

Given a sentence s, output its constituency parse tree.

$$\mathrm{parse}(s) = \arg\max_{\mathcal{Y}} \mathrm{score}(s, \mathcal{Y}; \boldsymbol{\Theta})$$

$\mathcal{Y}$: a parse tree.

$\boldsymbol{\Theta}$: model parameters.

score: a scoring function, e.g., the log probability of the parse tree assigned by a PCFG.

Constituency Parsing as an NLP Task

Given a sentence s, output its constituency parse tree.

$$\text{parse}(s) = \arg\max_{\mathcal{Y}} \text{score}(s, \mathcal{Y}; \boldsymbol{\Theta})$$

$\mathcal{Y}$: a parse tree.

$\boldsymbol{\Theta}$: model parameters.

score: a scoring function, e.g., the log probability of the parse tree assigned by a PCFG.

Most studies are based on the Penn Treebank (Marcus et al., 1993), led by researchers at UPenn.

Constituency Parsing as an NLP Task

Given a sentence s, output its constituency parse tree.

$$\text{parse}(s) = \arg\max_{\mathcal{Y}} \text{score}(s, \mathcal{Y}; \boldsymbol{\Theta})$$

$\mathcal{Y}$: a parse tree.

$\boldsymbol{\Theta}$: model parameters.

score: a scoring function, e.g., the log probability of the parse tree assigned by a PCFG.

Most studies are based on the Penn Treebank (Marcus et al., 1993), led by researchers at UPenn.

Treebank: corpus of annotated parse trees.

The CKY Algorithm

The Cocke–Kasami–Younger (CKY) algorithm is a dynamic programming algorithm for constituency parsing. Define

$$\begin{aligned}\text{parse}(s) &= \arg\max_{\mathcal{Y}} \text{score}(s, \mathcal{Y}; \boldsymbol{\Theta}) \\ \text{score}(s, \mathcal{Y}; \boldsymbol{\Theta}) &= \sum_{r \in \mathcal{Y}} \log P_{\boldsymbol{\Theta}}(r)\end{aligned}$$

The CKY Algorithm

The Cocke–Kasami–Younger (CKY) algorithm is a dynamic programming algorithm for constituency parsing. Define

$$\begin{aligned}\mathrm{parse}(s) &= \arg\max_{\mathcal{Y}} \mathrm{score}(s, \mathcal{Y}; \mathbf{\Theta})\\ \mathrm{score}(s, \mathcal{Y}; \mathbf{\Theta}) &= \sum_{r \in \mathcal{Y}} \log P_{\mathbf{\Theta}}(r)\end{aligned}$$

The algorithm gives the most probable parse tree for a sentence within polynomial time.

The CKY Algorithm

The Cocke–Kasami–Younger (CKY) algorithm is a dynamic programming algorithm for constituency parsing. Define

$$\text{parse}(s) = \arg\max_{\mathcal{Y}} \text{score}(s, \mathcal{Y}; \boldsymbol{\Theta})$$
$$\text{score}(s, \mathcal{Y}; \boldsymbol{\Theta}) = \sum_{r \in \mathcal{Y}} \log P_{\boldsymbol{\Theta}}(r)$$

The algorithm gives the most probable parse tree for a sentence within polynomial time.

Key idea: let $F[i, j, A]$ as the highest-possible score of a parse tree with non-terminal A spanning words i to j.

The CKY Algorithm

$$F[i,j,A] = \max_{\substack{A \to B\ C \\ k \in [i,j-1]}} \{F[i,k,B] + F[k+1,j,C] + \log P_{\Theta}(A \to B\ C)\}$$

Edge cases: $F[i,i,A] = \log P_{\Theta}(A \to w_i)$ for (pre-)terminal rules.

- i: start index.
- j: end index.
- A: non-terminal.
- B, C: non-terminals.
- k: split index.

The CKY Algorithm

$$F[i,j,A] = \max_{\substack{A \to B\ C \\ k \in [i, j-1]}} \{F[i,k,B] + F[k+1,j,C] + \log P_{\boldsymbol{\Theta}}(A \to B\ C)\}$$

Edge cases: $F[i,i,A] = \log P_{\boldsymbol{\Theta}}(A \to w_i)$ for (pre-)terminal rules.

- i: start index.
- j: end index.
- A: non-terminal.
- B, C: non-terminals.
- k: split index.

The parse tree can be reconstructed by backtracking.

The CKY Algorithm

$$F[i, j, A] = \max_{\substack{A \rightarrow B\ C \\ k \in [i, j-1]}} \{F[i, k, B] + F[k+1, j, C] + \log P_{\boldsymbol{\Theta}}(A \rightarrow B\ C)\}$$

Edge cases: $F[i, i, A] = \log P_{\boldsymbol{\Theta}}(A \rightarrow w_i)$ for (pre-)terminal rules.

- i: start index.
- j: end index.
- A: non-terminal.
- B, C: non-terminals.
- k: split index.

The parse tree can be reconstructed by backtracking.

Time complexity: $O(n^3|\mathcal{R}|)$, where n is the sentence length and $\mathcal{R}$ is the set of rules.

The CKY Algorithm

$$F[i,j,A] = \max_{\substack{A \to B\ C \\ k \in [i, j-1]}} \{F[i,k,B] + F[k+1,j,C] + \log P_{\mathbf{\Theta}}(A \to B\ C)\}$$

Edge cases: $F[i,i,A] = \log P_{\mathbf{\Theta}}(A \to w_i)$ for (pre-)terminal rules.

- i: start index.
- j: end index.
- A: non-terminal.
- B, C: non-terminals.
- k: split index.

The parse tree can be reconstructed by backtracking.

Time complexity: $O(n^3|\mathcal{R}|)$, where n is the sentence length and $\mathcal{R}$ is the set of rules.

Space complexity: $O(n^2|\mathcal{N}|)$, where $\mathcal{N}$ is the set of non-terminals.

Neural Constituency Parsing

Problem formulation: given an input sentence, we score all $n(n-1)/2$ possible spans for each non-terminal label, and use CKY to find the best-scoring parse tree.

Neural Constituency Parsing

Problem formulation: given an input sentence, we score all $n(n-1)/2$ possible spans for each non-terminal label, and use CKY to find the best-scoring parse tree.

Implementation: use neural encoder (e.g., Transformer) to encode the input span, do pooling to make it a fixed-dimensional vector, and predict the score for each non-terminal label with an MLP.

Neural Constituency Parsing

Problem formulation: given an input sentence, we score all $n(n-1)/2$ possible spans for each non-terminal label, and use CKY to find the best-scoring parse tree.

Implementation: use neural encoder (e.g., Transformer) to encode the input span, do pooling to make it a fixed-dimensional vector, and predict the score for each non-terminal label with an MLP.

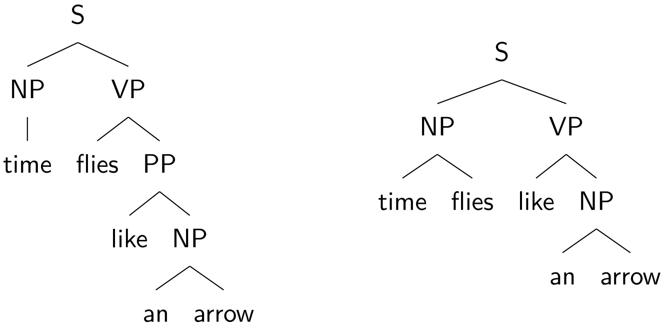
Training objective: encourage the ground-truth tree to have higher score than all other trees (see Kitaev and Klein, 2018).

$$\max_{\boldsymbol{\Theta}} \sum_{(s,\mathcal{Y})\in\mathcal{D}} \left(\sum_{(\ell,r)\in\mathcal{Y}} \text{score}(\ell, r; s, \boldsymbol{\Theta}) - \underbrace{\max_{\mathcal{Y}'} \sum_{(\ell,r)\in\mathcal{Y}'} \text{score}(\ell, r; s, \boldsymbol{\Theta})}_{\text{CKY algorithm}} \right)$$

Evaluation of Constituency Parsing

Bracketing F1 score: the harmonic mean of precision and recall of the bracketing.

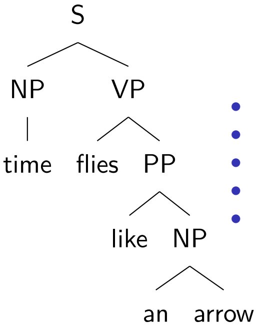
The evalb toolkit: `https://nlp.cs.nyu.edu/evalb/`.



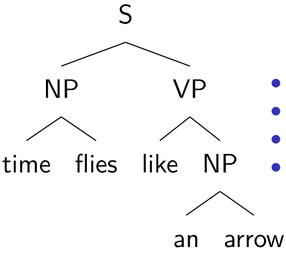
Evaluation of Constituency Parsing

Bracketing F1 score: the harmonic mean of precision and recall of the bracketing.

The evalb toolkit: `https://nlp.cs.nyu.edu/evalb/`.



- (S, 1, 5)
- (NP, 1, 1)
- (VP, 2, 5)
- (PP, 3, 5)
- (NP, 4, 5)

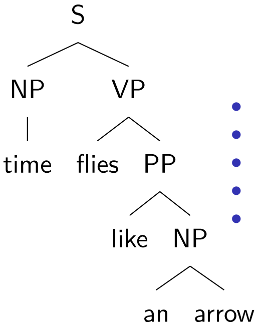


- (S, 1, 5)
- (NP, 1, 2)
- (VP, 3, 5)
- (NP, 4, 5)

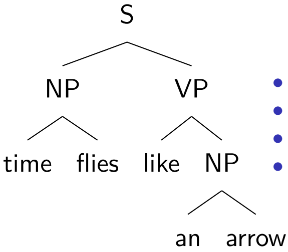
Evaluation of Constituency Parsing

Bracketing F1 score: the harmonic mean of precision and recall of the bracketing.

The evalb toolkit: `https://nlp.cs.nyu.edu/evalb/`.



- **(S, 1, 5)**
- (NP, 1, 1)
- (VP, 2, 5)
- (PP, 3, 5)
- **(NP, 4, 5)**

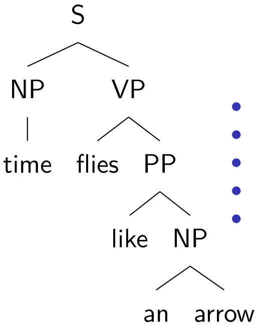


- **(S, 1, 5)**
- (NP, 1, 2)
- (VP, 3, 5)
- **(NP, 4, 5)**

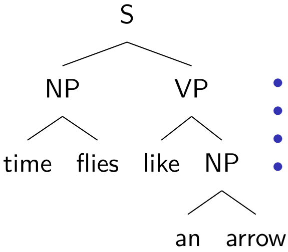
Evaluation of Constituency Parsing

Bracketing F1 score: the harmonic mean of precision and recall of the bracketing.

The evalb toolkit: `https://nlp.cs.nyu.edu/evalb/`.



- **(S, 1, 5)**
- (NP, 1, 1)
- (VP, 2, 5)
- (PP, 3, 5)
- **(NP, 4, 5)**



- **(S, 1, 5)**
- (NP, 1, 2)
- (VP, 3, 5)
- **(NP, 4, 5)**

$$\text{Precision} = \frac{2}{5} \quad \text{Recall} = \frac{2}{4} \quad F_1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = 0.44$$

Next

Syntax: Dependency Parsing