ECE59500NL Lecture 14: CKY

Jeffrey Mark Siskind

School of Electrical and Computer Engineering

Spring 2021

What makes a sentence grammatical?

Some sentences are grammatical:

> My brother likes pizza.
> The man in the red hat crossed the street.

Others are not:

> *John crossed the.
> *likes pizza.

Key questions:

- How can we *describe* what makes a sentence grammatical?
- How can we get a computer to *use* that description?

There are many grammatical sentences

Could try to list all grammatical sentences but...

My brother	*likes pizza.*
The man in the red hat	*crossed the street.*
The professor	*taught the class.*
The student	*ate pizza.*
⋮	⋮

There are many possible combinations, too many to list.
In fact there are an *infinite* number of grammatical sentences:

- *My father bought a goat for two zuzim.*
- *The cat chassed the goat that my father bought for two zuzim.*
- *The dog bit the cat that chassed the goat that my father bought for two zuzim.*
- *The stick beat the dog that bit the cat that chassed the goat that my father bought for two zuzim.*
- *The fire burned the stick that beat the dog that bit the cat that chassed the goat that my father bought for two zuzim.*

Key question:

- How can we describe an infinite number of grammatical sentences with a *finite* preferably *small* description?

Intuitive Idea

A sentence consists of a subject followed by a predicate.

Subject	**Predicate**
My brother	*likes pizza.*
The man in the red hat	*crossed the street.*
The professor	*taught the class.*
The student	*ate pizza.*
⋮	⋮

Can write this formally as:

$$\textit{Sentence} \rightarrow \textit{Subject Predicate}$$

or as:

$$\text{S} \rightarrow \text{NP VP}$$

Noun Phrases

The car
The big car
The big red car
The big bright red car
Every big bright red car
Some big bright red car
My big bright red car
The red car in the lot
The red car in the lot near my house
The Porsche in the lot near my house

$$
\begin{aligned}
\text{NP} &\rightarrow \text{D N}\\
\text{NP} &\rightarrow \text{D } \overline{\text{N}}\\
\overline{\text{N}} &\rightarrow \text{A } \overline{\text{N}}\\
\overline{\text{N}} &\rightarrow \text{A N}\\
\overline{\text{N}} &\rightarrow \overline{\text{N}} \text{ PP}\\
\overline{\text{N}} &\rightarrow \text{N PP}
\end{aligned}
$$

$$
\begin{aligned}
\text{D} &\rightarrow \textit{the}\\
\text{D} &\rightarrow \textit{every}\\
\text{D} &\rightarrow \textit{some}\\
\text{D} &\rightarrow \textit{my}\\
\text{A} &\rightarrow \textit{big}\\
\text{A} &\rightarrow \textit{red}\\
\text{A} &\rightarrow \textit{bright}\\
\text{N} &\rightarrow \textit{car}\\
\text{N} &\rightarrow \textit{lot}\\
\text{N} &\rightarrow \textit{house}\\
\text{N} &\rightarrow \textit{Porsche}
\end{aligned}
$$

Prepositional Phrases

in the lot
in the lot near my house

$$
\begin{aligned}
\text{PP} &\rightarrow \text{P NP} \\
\text{P} &\rightarrow \textit{in} \\
\text{P} &\rightarrow \textit{near}
\end{aligned}
$$

Verb Phrases

likes pizza
crossed the street
taught the class
ate pizza
crossed under the bridge
taught in the University
ate at the party

$$
\begin{aligned}
\text{VP} &\rightarrow \text{V NP}\\
\text{VP} &\rightarrow \text{V PP}\\
\text{V} &\rightarrow \textit{likes}\\
\text{V} &\rightarrow \textit{crossed}\\
\text{V} &\rightarrow \textit{taught}\\
\text{V} &\rightarrow \textit{ate}
\end{aligned}
$$

Grammar Summary

$$
\begin{array}{rcl}
\text{S} & \rightarrow & \text{NP VP} \\
\text{NP} & \rightarrow & \text{D } \overline{\text{N}} \\
\text{NP} & \rightarrow & \text{D N} \\
\overline{\text{N}} & \rightarrow & \text{A } \overline{\text{N}} \\
\overline{\text{N}} & \rightarrow & \text{A N} \\
\overline{\text{N}} & \rightarrow & \overline{\text{N}} \text{ PP} \\
\overline{\text{N}} & \rightarrow & \text{N PP} \\
\text{PP} & \rightarrow & \text{P NP} \\
\text{VP} & \rightarrow & \text{V NP} \\
\text{VP} & \rightarrow & \text{V PP}
\end{array}
$$

- Can interpret from left to right as a mechanism for *producing* sentences.
- Can interpret from right to left as a mechanism for *analyzing* sentences.

Generating Sentences

```
(define (generate-s)
 (string-append (generate-np) " " (generate-vp)))
(define (generate-np)
 (random-either
  (string-append (generate-d) " " (generate-nbar))
  (string-append (generate-d) " " (generate-n))))
(define (generate-nbar)
 (random-either
  (string-append (generate-a) " " (generate-nbar))
  (string-append (generate-a) " " (generate-n))
  (string-append (generate-nbar) " " (generate-pp))
  (string-append (generate-n) " " (generate-pp))))
(define (generate-pp)
  (string-append (generate-p) " " (generate-np)))
(define (generate-vp)
 (random-either
  (string-append (generate-v) " " (generate-np))
  (string-append (generate-v) " " (generate-pp))))
(define (generate-d)
 (random-either "the" "every" "some" "my"))
(define (generate-n)
 (random-either "car" "lot" "house" "Porsche"))
(define (generate-a)
 (random-either "big" "red" "bright"))
(define (generate-p)
 (random-either "in" "near"))
(define (generate-v)
 (random-either "likes" "taught" "crossed" "ate"))
```

What makes a sentence grammatical?

Some sentences are grammatical:

My brother likes pizza.
The man in the red hat crossed the street.

Others are not:

*John crossed the.
*likes pizza.

Key questions:

- How can we *describe* what makes a sentence grammatical?
- How can we get a computer to *use* that description?

Testing whether a sentence is grammatical—I

```
(define (can-be-s? x)
 (some-split?
  (lambda (y z)
   (and (can-be-np? y) (can-be-vp? z)))
  x))
(define (can-be-np? x)
 (some-split?
  (lambda (y z)
   (and (can-be-d? y)
        (or (can-be-nbar? z) (can-be-n? z))))
  x))
(define (can-be-nbar? x)
 (some-split?
  (lambda (y z)
   (or (and (can-be-a? y)
            (or (can-be-nbar? z) (can-be-n? z)))
       (and (or (can-be-nbar? y) (can-be-n? y))
            (can-be-pp? z))))
  x))
```

Testing whether a sentence is grammatical—II

```
(define (can-be-pp? x)
 (some-split?
  (lambda (y z)
   (and (can-be-p? y) (can-be-np? z)))
  x))
(define (can-be-vp? x)
 (some-split?
  (lambda (y z)
   (and (can-be-v? y)
        (or (can-be-np? z) (can-be-pp? z))))
  x))
(define (can-be-d? x)
 (member x '("the" "every" "some" "my")))
(define (can-be-n? x)
 (member x '("car" "lot" "house" "Porsche")))
(define (can-be-a? x)
 (member x '("big" "red" "bright")))
(define (can-be-p? x)
 (member x '("in" "near")))
(define (can-be-v? x)
 (member x '("likes" "taught" "crossed" "ate")))
```

Why the naive approach is bad

The man in the car in the street near the bridge ate pizza.

A better approach

$_0$ *The* $_1$ *man* $_2$ *ate* $_3$ *pizza* $_4$

Each entry of $M[i,j]$ will store a subset of $\{\mathrm{S}, \mathrm{NP}, \overline{\mathrm{N}}, \mathrm{PP}, \mathrm{VP}, \mathrm{D}, \mathrm{N}, \mathrm{A}, \mathrm{P}, \mathrm{V}\}$

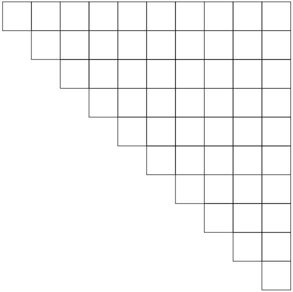
$c \in M[i,j]$ iff the substring from i to j can be a phrase of type c

CKY Algorithm

Initialize $M[i, i+1]$ to the category of word i.

$$M[i,j] \leftarrow \{A | A \rightarrow B\ C \wedge i < k < j \wedge B \in M[i,k] \wedge C \in M[k,j]\}$$

CKY algorithm



$$
\begin{array}{rcl}
\text{S} & \rightarrow & \text{NP VP} \\
\text{NP} & \rightarrow & \text{D } \overline{\text{N}} \\
\text{NP} & \rightarrow & \text{D N} \\
\overline{\text{N}} & \rightarrow & \text{A } \overline{\text{N}} \\
\overline{\text{N}} & \rightarrow & \text{A N} \\
\overline{\text{N}} & \rightarrow & \overline{\text{N}} \text{ PP} \\
\overline{\text{N}} & \rightarrow & \text{N PP} \\
\text{PP} & \rightarrow & \text{P NP} \\
\text{VP} & \rightarrow & \text{V NP} \\
\text{VP} & \rightarrow & \text{V PP}
\end{array}
$$

Parse Trees

Every pawn is on some square.