CFGS – TAKE 2

David Kauchak
CS30 – Spring 2016

Admin

Lab/neural net package?

Assignment 6

Grammars

Language view:

A grammar is a set of structural rules that govern the composition of sentences, phrases and words.

Computational view:

A grammar (often called a "formal grammar") is a set of rules that describe what strings are valid in a formal language.

CFG production rules

$S \rightarrow NP\ VP$

left hand side (single symbol)

right hand side (one or more symbols)

CFG example

S → A B C
A → I
B → really
B → really, B
C → like cs

CFGs formally

G = (**NT**, **T**, **P**, **S**)

NT: finite set of nonterminal symbols

T: finite set of terminal symbols, **NT** and **T** are disjoint

P: finite set of productions of the form
$A \rightarrow \alpha$, $A \in NT$ and $\alpha \in (T \cup NT)^*$

S ∈ NT: start symbol

CFG example

Grammars "generate" or "derive" strings:

S → A B C
A → I
B → really
B → really, B
C → like cs

S

CFG example

Grammars "generate" or "derive" strings:

S → A B C
A → I
B → really
B → really, B
C → like cs

S

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right

CFG example

Grammars "generate" or "derive" strings:

S → A B C
A → I
B → really
B → really, B
C → like cs

A B C

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right

CFG example

Grammars "generate" or "derive" strings:

S → A B C
A → I
B → really
B → really, B
C → like cs

A B C

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right

CFG example

Grammars "generate" or "derive" strings:

S → A B C
A → I
B → really
B → really, B
C → like cs

A really C

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right

CFG example

Grammars "generate" or "derive" strings:

S → A B C
A → I
B → really
B → really, B
C → like cs

A really C

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right

CFG example

Grammars "generate" or "derive" strings:

S → A B C
A → I
B → really
B → really, B
C → like cs

A really like cs

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right

CFG example

Grammars "generate" or "derive" strings:

S → A B C
A → I
B → really
B → really, B
C → like cs

A really like cs

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right

CFG example

Grammars "generate" or "derive" strings:

S → A B C
A → I
B → really
B → really, B
C → like cs

I really like cs

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right

CFG example

Grammars "generate" or "derive" strings:

S → A B C
A → I
B → really
B → really, B
C → like cs

I really like cs

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right

No more rules apply, so we're done!

CFG example

S → A B C
A → I
B → really
B → really, B
C → like cs

Grammars "generate" or "derive" strings:

I really like cs

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right

CFG example

S → A B C
A → I
B → really
B → really, B
C → like cs

Grammars "generate" or "derive" strings:

A really, B C

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right

CFG example

S → A B C
A → I
B → really
B → really, B
C → like cs

Grammars "generate" or "derive" strings:

A really, really, B C

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right

CFG example

S → A B C
A → I
B → really
B → really, B
C → like cs

Grammars describe a language, i.e. the strings (aka sentences) that are part of that language

I really, really, … like cs

What language does this represent?

S → aS
S → E
E → bE
E → b

What language does this represent?

S → aS
S → E
E → bE
E → b

Two options

S

What language does this represent?

S → aS
S → E
E → bE
E → b

S
↓
aS

What language does this represent?

S → aS
S → E
E → bE
E → b

aS
↓
aaS

What language does this represent?

S → aS
S → E
E → bE
E → b

aaS
↓
aaaS

- Can do this as many times as we want
- Keeps adding more a's to the front

What language does this represent?

S → aS
S → E
E → bE
E → b

aaaS
↓
aaaE

Eventually, apply second rule

What language does this represent?

S → aS
S → E
E → bE
E → b

Two options

aaaE

What language does this represent?

S → aS
S → E
E → bE
E → b

aaaE
↓
aaabE

What language does this represent?

S → aS
S → E
E → bE
E → b

aaabE
↓
aaabbE

What language does this represent?

S → aS
S → E
E → bE
E → b

aaabbE
↓
aaabbbE

What language does this represent?

S → aS
S → E
E → bE
E → b

aaabbE
↓
aaabb…bE

- Can do this as many times as we want
- Keeps adding more b's to the end

What language does this represent?

S → aS
S → E
E → bE
E → b

aaabb…bE
↓
aaabb…bb

Eventually, apply second rule

What language does this represent?

S → aS
S → E
E → bE
E → b

aaabb…bE
↓
aaabb…bb

Grammar represents all strings with zero or more a's followed by one or more b's

Notational convenience

S → aS
S → E
E → bE
E → b

S → aS | E
E → bE | b

Often many ways to write the same language

S → aS | E
E → bE | b

S → aS | E
E → Eb | b

S → aS | aaS | E
E → Eb | b

What languages do these represent?

S → aEa | bEb
E → Ea | Eb | a | b

S → aSb
S → ab

S → aaS | abS | baS | bbS | ε

nothing

What languages do these represent?

S → aEa | bEb
E → Ea | Eb | a | b

all strings of a's and b's that start and end with the same letter

S → aSb
S → ab

strings of a's followed by an equal number of b's

S → aaS | abS | baS | bbS | ε

all strings of a's and b's with even length

Writing CFGs

Write a CFG to represent the language containing all strings that start with a.

S → aT
T → Ta | Tb | ε

Writing CFGs

Write a CFG to represent the language containing all strings with exactly two bs.

S → TbTbT
T → Ta | ε

CFG: Another example

Many possible CFGs for English, here is an example (fragment):

S → NP VP
VP → V NP
NP → DetP N | DetP AdjP N
AdjP → Adj | Adv AdjP
N → boy | girl
V → sees | likes
Adj → big | small
Adv → very
DetP → a | the

Derivations in a CFG

S → NP VP
VP → V NP
NP → DetP N | DetP AdjP N
AdjP → Adj | Adv AdjP
N → boy | girl
V → sees | likes
Adj → big | small
Adv → very
DetP → a | the

S

What can we do?

Derivations in a CFG

S → NP VP
VP → V NP
NP → DetP N | DetP AdjP N
AdjP → Adj | Adv AdjP
N → boy | girl
V → sees | likes
Adj → big | small
Adv → very
DetP → a | the

S

Derivations in a CFG

S → NP VP
VP → V NP
NP → DetP N | DetP AdjP N
AdjP → Adj | Adv AdjP
N → boy | girl
V → sees | likes
Adj → big | small
Adv → very
DetP → a | the

NP VP

What can we do?

Derivations in a CFG

S → NP VP
VP → V NP
NP → DetP N | DetP AdjP N
AdjP → Adj | Adv AdjP
N → boy | girl
V → sees | likes
Adj → big | small
Adv → very
DetP → a | the

NP VP

Derivations in a CFG

S → NP VP
VP → V NP
NP → DetP N | DetP AdjP N
AdjP → Adj | Adv AdjP
N → boy | girl
V → sees | likes
Adj → big | small
Adv → very
DetP → a | the

DetP N VP

Derivations in a CFG

S → NP VP
VP → V NP
NP → DetP N | DetP AdjP N
AdjP → Adj | Adv AdjP
N → boy | girl
V → sees | likes
Adj → big | small
Adv → very
DetP → a | **the**

DetP N VP

Derivations in a CFG

S → NP VP
VP → V NP
NP → DetP N | DetP AdjP N
AdjP → Adj | Adv AdjP
N → boy | girl
V → sees | **likes**
Adj → big | small
Adv → very
DetP → a | the

the boy VP

Derivations in a CFG

S → NP VP
VP → V NP
NP → DetP N | DetP AdjP N
AdjP → Adj | Adv AdjP
N → boy | **girl**
V → sees | likes
Adj → big | small
Adv → very
DetP → a | the

the boy likes NP

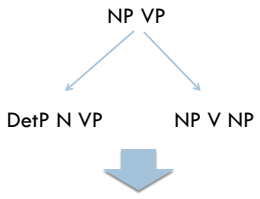
Derivations in a CFG

S → NP VP
VP → V NP
NP → DetP N | DetP AdjP N
AdjP → Adj | Adv AdjP
N → boy | girl
V → sees | likes
Adj → big | small
Adv → very
DetP → a | the

the boy likes a girl

Derivations in a CFG: Order of Derivation Irrelevant

S → NP VP
VP → V NP
NP → DetP N | DetP AdjP N
AdjP → Adj | Adv AdjP
N → boy | girl
V → sees | likes
Adj → big | small
Adv → very
DetP → a | the

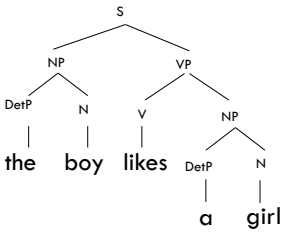


the boy likes a girl

Derivations of CFGs

Derivation history shows a tree:

the boy likes a girl



Another CFG example

S → NP VP
VP → V | V ADV
NP → ART PreNP
PreNP → N | ADJ PreNP
ADV → furiously | soothingly | intentionally
ADJ → colorless | green | smelly
ART → the | a
V → sleeps | eats | swims | sprints
N → idea | bagel | milk | cow

What can we generate?

One last example

S → N
S → (S)
S → S + S | S - S
S → S * S | S / S
N → 0 | 1 | 2 | … | 9
N → N N

What language does this CFG represent?

One last example

S → N
S → (S)
S → S + S | S - S
S → S * S | S / S
N → 0 | 1 | 2 | … | 9
N → N N

All arithmetic expressions!

Parsing

Given a CFG and a sentence, determine the possible parse tree(s)

S -> NP VP
NP -> N
NP -> PRP
NP -> N PP
VP -> V NP
VP -> V NP PP
PP -> IN N
PRP -> I
V -> eat
N -> sushi
N -> tuna
IN -> with

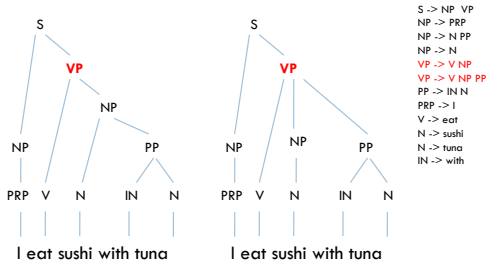
I eat sushi with tuna

What parse trees are possible for this sentence?

How did you do it?

What if the grammar is much larger?

Parsing



What is the difference between these parses?

CFGs implemented