Syntax: A Formal Introduction

Weiwei Sun

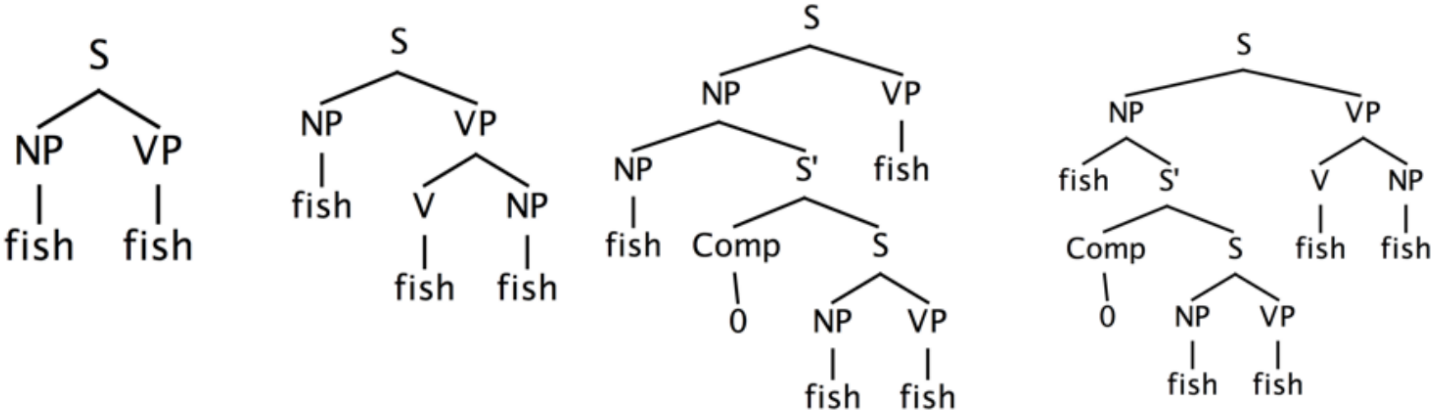
Institute of Computer Science and Technology
Peking University

September 26, 2017

Last lecture

Let's draw trees!

Fish fish
Fish fish fish
Fish fish fish fish
Fish fish fish fish fish



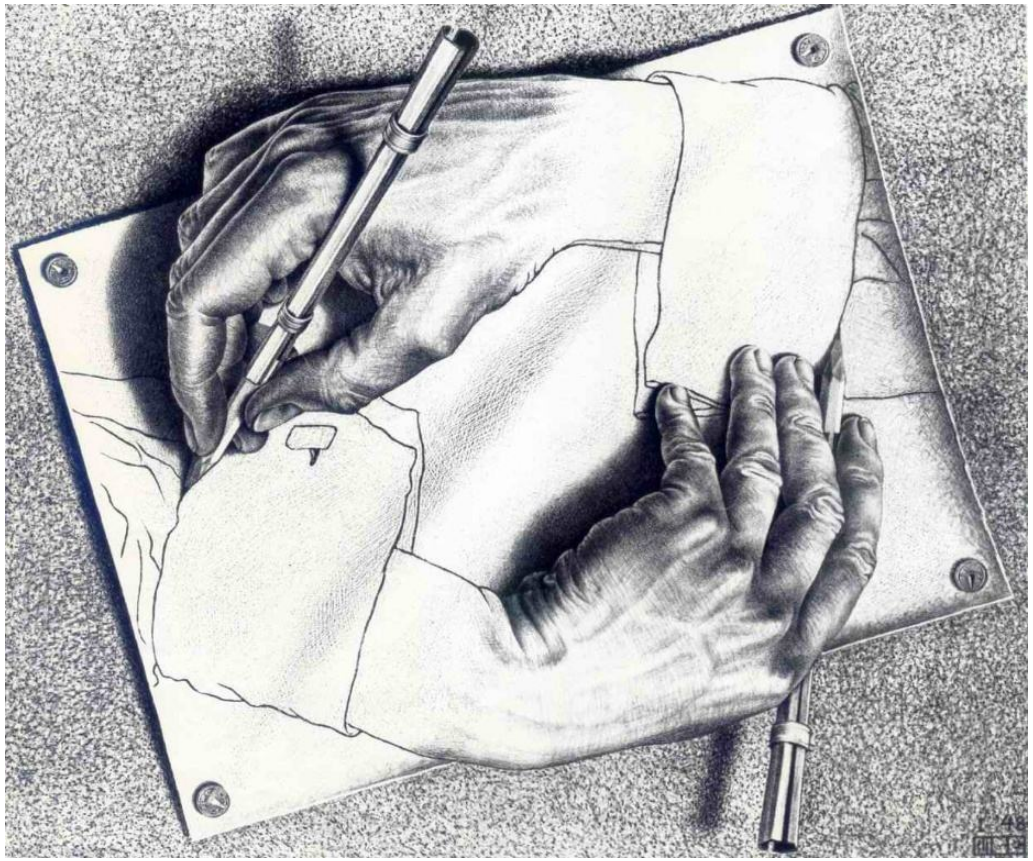
How to draw a tree?

Andrew Carnie. *Syntax: A Generative Introduction*

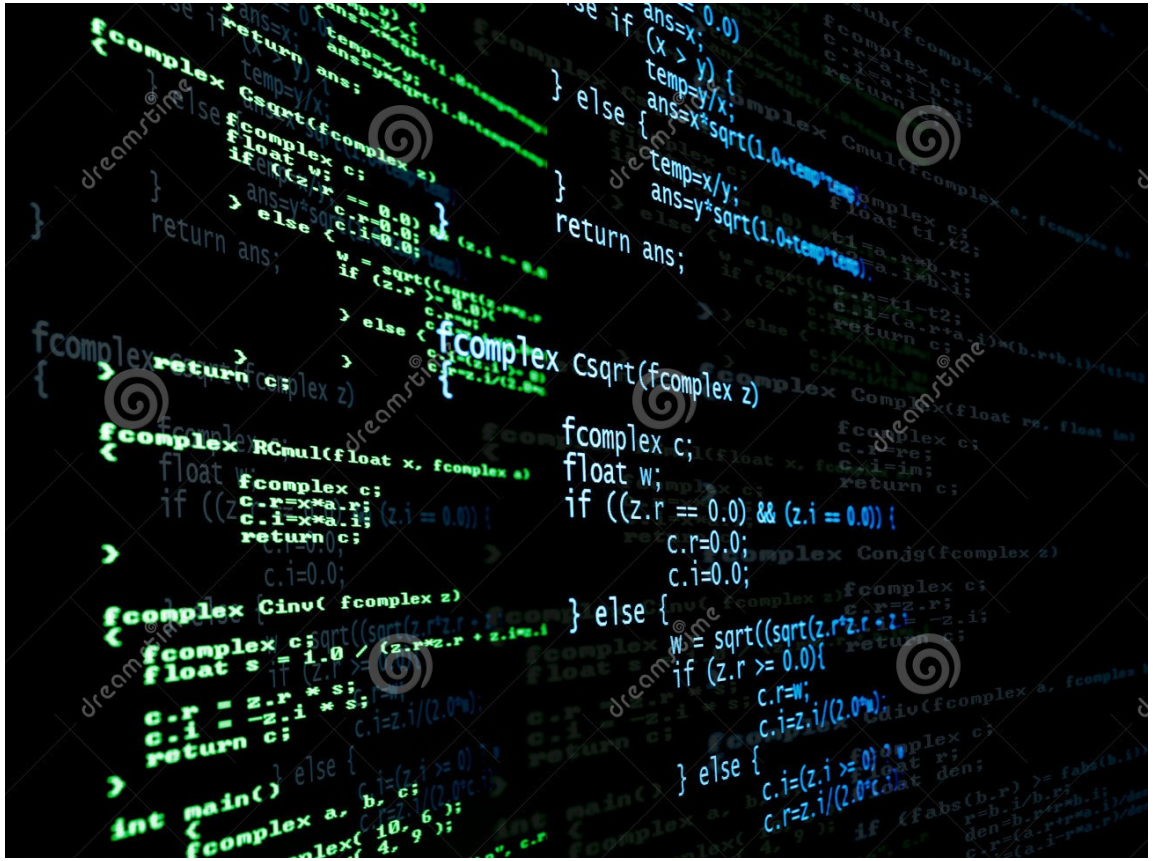
How to draw a tree?

How to draw a tree?

How to draw a tree?



How to draw a tree?



How to draw a tree?

Brackets

```
( (S (NP (NNP Ms.) (NNP Haag) )
     (VP (VBZ plays)
         (NP (NNP Elianti) ))
     (. .) ))
```

A simple js program

www.icst.pku.edu.cn/lcwm/course/fs/tree.html

How to draw a tree?

Brackets

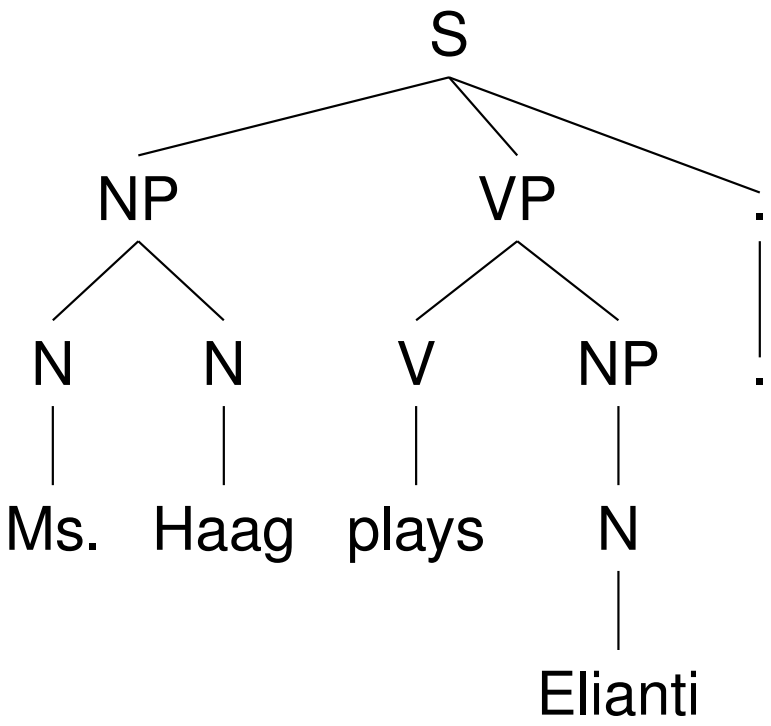
```
( (S (NP (NNP Ms.) (NNP Haag) )
     (VP (VBZ plays)
         (NP (NNP Elianti) ))
     (. .) ))
```

A simple js program

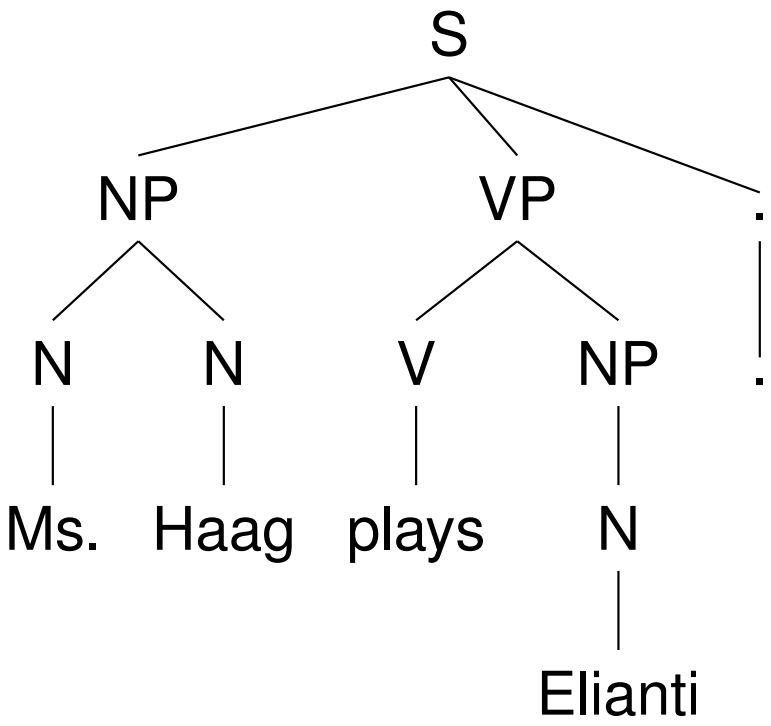
www.icst.pku.edu.cn/lcwm/course/fs/tree.html

(TOP (S (NP-SBJ (NNP Ms.) (NNP Haag)) (VP (VBZ plays) (NP (NNP Elianti))) (. .)))

What information does a tree encode?



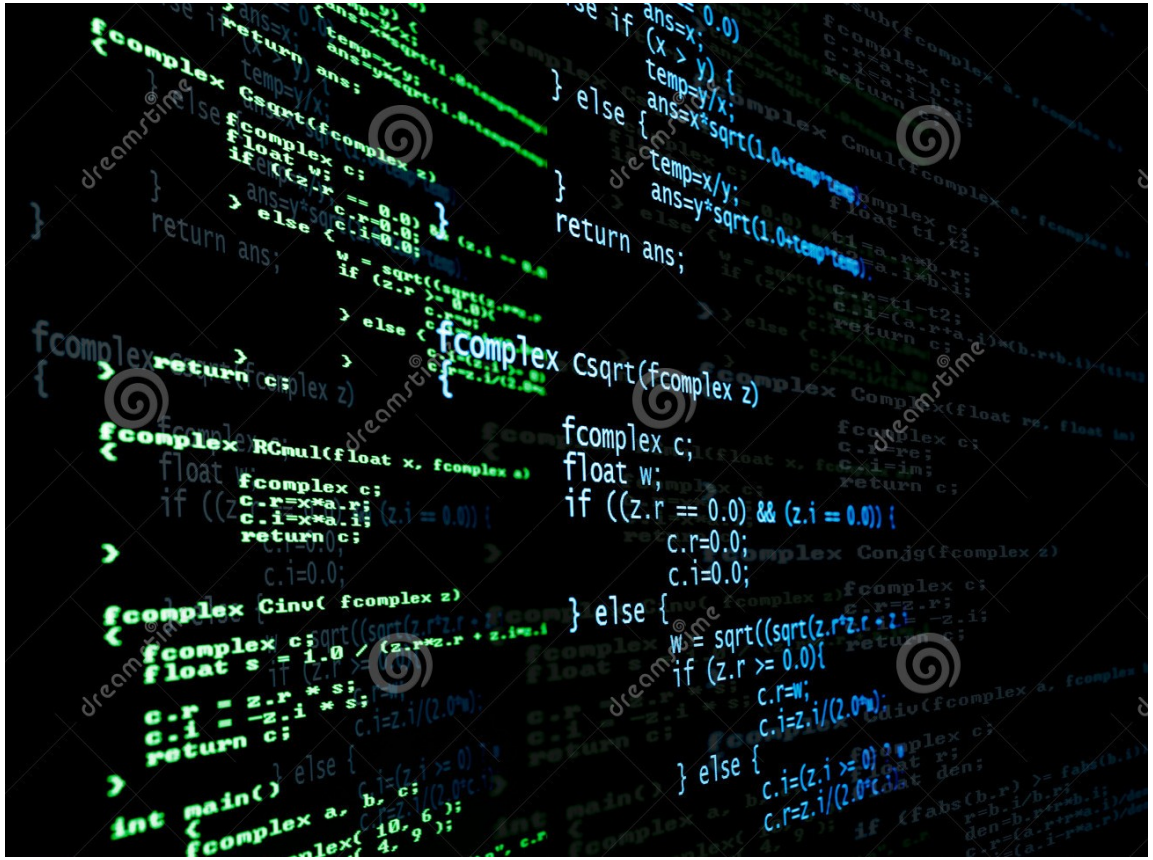
What information does a tree encode?



- Constituents
- Categories
- Structural relations, e.g. c-command

How to draw a tree?

Seriously, how to draw a tree?



For computation, you must provide precise definitions.

Comprehensive Theory or Model

- Phenomenon involving human behavior is extremely complex.
- Instead of a comprehensive theory, we devise a *model* that simplifies the phenomenon to capture some key aspect of it

What might we use a model for?

- Prediction: estimating the behavior/properties of a new state/datum on the basis of an existing dataset
- Hypothesis testing: as a framework for determining whether a given factor has an appreciable influence on some other variable
- Insight: Most generally, a good model can be explored in ways that give insight into the phenomenon under consideration

Syntax: What does it mean?

We can view syntax/syntactic theory in a number of ways, two of which are the following:

- ... model: ...
- Computational model: syntactic structures are formal objects which can be mathematically treated/manipulated

Syntax: What does it mean?

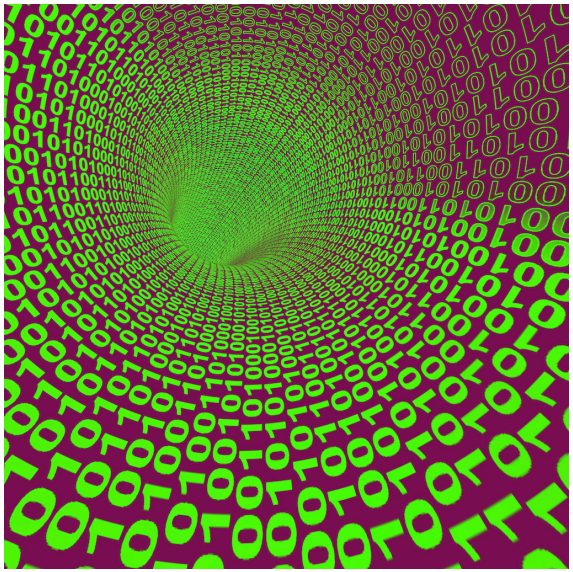
We can view syntax/syntactic theory in a number of ways, two of which are the following:

- ... model: ...
- Computational model: syntactic structures are formal objects which can be mathematically treated/manipulated

Syntax: What does it mean?

We can view syntax/syntactic theory in a number of ways, two of which are the following:

- ... model: ...
- Computational model: syntactic structures are formal objects which can be mathematically treated/manipulated



Syntax: What does it mean?

We can view syntax/syntactic theory in a number of ways, two of which are the following:

- ... model: ...
- Computational model: syntactic structures are formal objects which can be mathematically treated/manipulated

Syntax attempts to capture the nature of rules with which we generate

- strings of those words (weak generative power)
- structures which license strings of those words (strong generative power)

The generative revolution

- Writings on grammar go back at least 3000 years
- Until about 1800, almost all linguistics was primarily prescriptive, attempting to codify the *correct* way of talking.
- Discovery in 19th century: There was a historical connection among most of the languages of Europe, as well as Sanskrit and other languages of India (plus some languages in between).
  - ⇒ Historical linguistics: reconstructing the family tree of the Indo-European languages
  - ⇒ Syntactic comparison and reconstruction was also initiated.
- Early 20th century: Many linguists, following Ferdinand de Saussure, turned their attention from the *diachronic* study to the *synchronic* analysis.

Brief history


不欽玩耽味爲之義訓璞不揆擣昧少
習焉沈研鑽極二九載矣雖註者十餘
猶未詳備並多紛謬有所漏略是以復
集異聞會粹舊說考方國之語采謠俗
志錯綜樊孫悖闕羣言剗其瑕礫搴其
稂事有隱滯援據徵之其所易了闕而
論別爲音圖用袂未寤輒復擁篲清道
望塵躅者以將來君子爲亦有涉乎此
自序

爾雅卷上

釋詁第一　釋言第二　釋訓第三
釋親第四

釋詁第一　郭璞注

初哉首基肇祖元胎俶落權輿始也尚書曰三月惟
魏詩曰令終有
林烝天帝皇王后辟公侯君也詩曰有
弘鄭宏溥介純夏憮尾墳嘏丕奕洪
戎駿假京碩濯訏宇穹壬路遥甫景廢壯冢簡
昄晜將業蓄大也詩曰我受命溥將又曰乾如此
駿厖鴻孫奏假王公伊濯訏謨定命右

The generative revolution

- Ferdinand de Saussure
- Towards modern syntax
  - Structuralism (1920s-30s): Bloomfield
  - Distributionalism (1950s): Hockett, Harris
  - Categorial grammar (1930s): Adjukiewicz
  - Dependency grammar (1930s): Tesnière
- Noam Chomsky's work in the 1950s radically changed linguistics, making syntax central.

The generative revolution

Chomsky (*Syntactic Structures*)

By pushing a precise but inadequate formulation to an unacceptable conclusion, we can often expose the exact source of this inadequacy and, consequently, gain a deeper understanding of the linguistic data. [...] Obscure and intuition-bound notions can neither lead to absurd conclusions nor provide new and correct ones, [...]

The generative revolution

Chomsky (*Syntactic Structures*)

By pushing a precise but inadequate formulation to an unacceptable conclusion, we can often expose the exact source of this inadequacy and, consequently, gain a deeper understanding of the linguistic data. [...] Obscure and intuition-bound notions can neither lead to absurd conclusions nor provide new and correct ones, [...]

Main tenets of Generative Grammar

- Grammars should be formulated precisely and explicitly.
- The theory of grammar is a theory of human linguistic abilities.

The generative revolution

Chomsky (*Aspects of the Theory of Syntax*)

A grammar of a language purports to be a description of the ideal speaker-hearer's intrinsic competence. If the grammar is, furthermore, perfectly explicit—in other words, if it does not rely on the intelligence of the understanding reader but rather provides an explicit analysis of his contribution—we may (somewhat redundantly) call it a generative grammar.

Main tenets of Generative Grammar

- Grammars should be formulated precisely and explicitly.
- The theory of grammar is a theory of human linguistic abilities.

The generative revolution

Chomsky (*Aspects of the Theory of Syntax*)

A grammar of a language purports to be a description of the ideal speaker-hearer's intrinsic competence. If the grammar is, furthermore, perfectly explicit—in other words, if it does not rely on the intelligence of the understanding reader but rather provides an explicit analysis of his contribution—we may (somewhat redundantly) call it a generative grammar.

Main tenets of Generative Grammar

- Grammars should be formulated precisely and explicitly.
- The theory of grammar is a theory of human linguistic abilities.

The generative revolution

Chomsky (*Aspects of the Theory of Syntax*)

A grammar of a language purports to be a description of the ideal speaker-hearer's intrinsic competence. If the grammar is, furthermore, perfectly explicit—in other words, if it does not rely on the intelligence of the understanding reader but rather provides an explicit analysis of his contribution—we may (somewhat redundantly) call it a generative grammar.

Main tenets of Generative Grammar

- Grammars should be formulated precisely and explicitly.
- The theory of grammar is a theory of human linguistic abilities.

The generative revolution

Chomsky (*Aspects of the Theory of Syntax*)

A grammar of a language purports to be a description of the ideal speaker-hearer's intrinsic competence. If the grammar is, furthermore, perfectly explicit—in other words, if it does not rely on the intelligence of the understanding reader but rather provides an explicit analysis of his contribution—we may (somewhat redundantly) call it a generative grammar.

Chomsky (*The Minimalist Program*)

I have always understood a generative grammar to be nothing more than an explicit grammar.

Descriptive adequacy

Goal

- Providing accurate structural descriptions for well-formed sentences
- Giving an explicit encoding of a language
- Approaching broad coverage, i.e., aiming to describe all of the well-formed sentences possible in a language

Circularity

How can be state the usage of a word except in other words?

★ We use a **metalanguage**.

Precise encoding

Last lecture: How to DO syntax?

At the core of syntactic investigation are the rules that allow certain elements to be combined. To explore such rules, we need to study:

- the atomic elements which are the input of the initial combination
- the demarcation of the units in a sentence
- the structural relationship between the units

Precise encoding

Given a formal way to **generate** sets of strings or structures by precisely defining:

- elementary structures
- ways of combining those structures

Such an emphasis on mathematical precision makes these grammar formalism more easily implementable

Phrase structure grammar

- The formalism of context-free grammars was developed in the mid-1950s by Noam Chomsky.
- Phrase structure grammars are also known as constituency grammars.
- There are probably languages that cannot be described by a context-free grammar (CFG)
  - Shown in the 1980s to be correct, for at least for Swiss German
  - English may be within the descriptive power of a CFG
  - But there may be other reasons beyond formal power to reject CFGs for representing natural languages ...
- Account for the tree-like structure that sentences have.

Definition of Context-Free Grammars

Four components in a grammatical description of a language:

1. A finite set of symbols that form the strings of a language.
   - We call this alphabet the terminals or terminal symbols.
   - In terms of syntactic analysis, this alphabet is the **lexicon**.
2. A finite set of variables, also called nonterminals or syntactic categories.
   - Each variable represents a class of strings, i.e., a set of strings.
3. START symbol: One of the variables represents the language being defined.
   - Other variables represent auxiliary classes of strings that are used to help define the language.
4. A finite set of productions or rules that represent the recursive definition of a language. Each production consists of:
   - A variable h
   - The production symbol $\rightarrow$
   - A string of zero or more terminals and variables. This string represents one way to form strings in the class of h.
     - Leave terminals unchanged
     - Substitute each variable with any string in it.

Comprehensive theory or model

Instead of a comprehensive theory, we devise a *model* that simplifies the phenomenon to capture some key aspect of it.

What is a language?

Language is a collection of sentences.

$$\mathcal{L} = \left\{ \begin{array}{l} \text{Colorless green ideas sleep furiously,} \\ \text{Colorless ideas sleep furiously,} \\ \text{Green ideas sleep furiously,} \\ \ldots \end{array} \right\}$$

Definition of Context-Free Grammars

The four components form a context-free grammar. We represent a CFG G by its four components, $G = (N, \Sigma, P, S)$.

1. N: nonterminals
2. Σ: terminals
3. P: productions
4. S: START

Question

What does *context-free* mean?

An example

1. $N = \{\text{S}, \text{NP}, \text{VP}, \text{AdjP}, \text{AdvP}\} \cup \{\text{N}_{pl}, \text{V}, \text{Adj}, \text{Adv}\}$
2. $\Sigma = \{$colorless, green, ideas, sleep, furiously$\}$
3. P

$\text{S} \rightarrow \text{NP VP}$	$\text{NP} \rightarrow \text{AdjP NP}$
$\text{VP} \rightarrow \text{VP AdvP}$	
$\text{VP} \rightarrow \text{V}$	$\text{NP} \rightarrow \text{N}_{pl}$
$\text{AdvP} \rightarrow \text{Adv}$	$\text{AdjP} \rightarrow \text{Adj}$

$\text{Adj} \rightarrow$ colorless	$\text{Adj} \rightarrow$ green
$\text{N}_{pl} \rightarrow$ ideas	$\text{V} \rightarrow$ sleep
$\text{Adv} \rightarrow$ furiously	

4. S

An example

1. $N = \{\text{S}, \text{NP}, \text{VP}, \text{AdjP}, \text{AdvP}\} \cup \{\text{N}_{pl}, \text{V}, \text{Adj}, \text{Adv}\}$
2. $\Sigma = \{$colorless, green, ideas, sleep, furiously$\}$
3. P

$\text{S} \rightarrow \text{NP VP}$	$\text{NP} \rightarrow \text{AdjP NP}$
$\text{VP} \rightarrow \text{VP AdvP}$	
$\text{VP} \rightarrow \text{V}$	$\text{NP} \rightarrow \text{N}_{pl}$
$\text{AdvP} \rightarrow \text{Adv}$	$\text{AdjP} \rightarrow \text{Adj}$

$\text{Adj} \rightarrow$ colorless	$\text{Adj} \rightarrow$ green
$\text{N}_{pl} \rightarrow$ ideas	$\text{V} \rightarrow$ sleep
$\text{Adv} \rightarrow$ furiously	

4. S

We can derive the structure of a string.

- **S**

An example

1. $N = \{\text{S}, \text{NP}, \text{VP}, \text{AdjP}, \text{AdvP}\} \cup \{\text{N}_{pl}, \text{V}, \text{Adj}, \text{Adv}\}$
2. $\Sigma = \{$colorless, green, ideas, sleep, furiously$\}$
3. P

S $\rightarrow$ NP VP	NP $\rightarrow$ AdjP NP
VP $\rightarrow$ VP AdvP	
VP $\rightarrow$ V	NP $\rightarrow$ N_{pl}
AdvP $\rightarrow$ Adv	AdjP $\rightarrow$ Adj

Adj $\rightarrow$ colorless	Adj $\rightarrow$ green
N_{pl} $\rightarrow$ ideas	V $\rightarrow$ sleep
Adv $\rightarrow$ furiously	

4. S

We can derive the structure of a string.

- S$\Rightarrow$ **NP VP**

An example

1. $N = \{\text{S}, \text{NP}, \text{VP}, \text{AdjP}, \text{AdvP}\} \cup \{\text{N}_{pl}, \text{V}, \text{Adj}, \text{Adv}\}$
2. $\Sigma = \{$colorless, green, ideas, sleep, furiously$\}$
3. P

S $\rightarrow$ NP VP	NP $\rightarrow$ AdjP NP
VP $\rightarrow$ VP AdvP	
VP $\rightarrow$ V	NP $\rightarrow$ N_{pl}
AdvP $\rightarrow$ Adv	AdjP $\rightarrow$ Adj

Adj $\rightarrow$ colorless	Adj $\rightarrow$ green
N_{pl} $\rightarrow$ ideas	V $\rightarrow$ sleep
Adv $\rightarrow$ furiously	

4. S

We can derive the structure of a string.

- S$\Rightarrow$ NP VP
  $\Rightarrow$ $\mathbf{N}_{pl}$ **VP**

An example

1. $N = \{\text{S}, \text{NP}, \text{VP}, \text{AdjP}, \text{AdvP}\} \cup \{\text{N}_{pl}, \text{V}, \text{Adj}, \text{Adv}\}$
2. $\Sigma = \{$colorless, green, ideas, sleep, furiously$\}$
3. P

S $\rightarrow$ NP VP	NP $\rightarrow$ AdjP NP
VP $\rightarrow$ VP AdvP	
VP $\rightarrow$ V	NP $\rightarrow$ N_{pl}
AdvP $\rightarrow$ Adv	AdjP $\rightarrow$ Adj

Adj $\rightarrow$ colorless	Adj $\rightarrow$ green
N_{pl} $\rightarrow$ ideas	V $\rightarrow$ sleep
Adv $\rightarrow$ furiously	

4. S

We can derive the structure of a string.

- S$\Rightarrow$ NP VP
  $\Rightarrow$ N_{pl} VP
  $\Rightarrow$ **ideas VP**

An example

1. $N = \{\text{S}, \text{NP}, \text{VP}, \text{AdjP}, \text{AdvP}\} \cup \{\text{N}_{pl}, \text{V}, \text{Adj}, \text{Adv}\}$
2. $\Sigma = \{$colorless, green, ideas, sleep, furiously$\}$
3. P

S $\rightarrow$ NP VP	NP $\rightarrow$ AdjP NP
VP $\rightarrow$ VP AdvP	
VP $\rightarrow$ V	NP $\rightarrow$ N_{pl}
AdvP $\rightarrow$ Adv	AdjP $\rightarrow$ Adj

Adj $\rightarrow$ colorless	Adj $\rightarrow$ green
N_{pl} $\rightarrow$ ideas	V $\rightarrow$ sleep
Adv $\rightarrow$ furiously	

4. S

We can derive the structure of a string.

- S$\Rightarrow$ NP VP
  $\Rightarrow$ N_{pl} VP
  $\Rightarrow$ ideas VP
  $\Rightarrow$ **ideas V AdvP**

An example

1. $N = \{\text{S}, \text{NP}, \text{VP}, \text{AdjP}, \text{AdvP}\} \cup \{\text{N}_{pl}, \text{V}, \text{Adj}, \text{Adv}\}$
2. $\Sigma = \{$colorless, green, ideas, sleep, furiously$\}$
3. P

S $\rightarrow$ NP VP	NP $\rightarrow$ AdjP NP
VP $\rightarrow$ VP AdvP	
VP $\rightarrow$ V	NP $\rightarrow$ N_{pl}
AdvP $\rightarrow$ Adv	AdjP $\rightarrow$ Adj

Adj $\rightarrow$ colorless	Adj $\rightarrow$ green
N_{pl} $\rightarrow$ ideas	V $\rightarrow$ sleep
Adv $\rightarrow$ furiously	

4. S

We can derive the structure of a string.

- S$\Rightarrow$ NP VP
  $\Rightarrow$ N_{pl} VP
  $\Rightarrow$ ideas VP
  $\Rightarrow$ ideas V AdvP
  $\Rightarrow$ **ideas sleep AdvP**

An example

1. $N = \{\text{S}, \text{NP}, \text{VP}, \text{AdjP}, \text{AdvP}\} \cup \{\text{N}_{pl}, \text{V}, \text{Adj}, \text{Adv}\}$
2. $\Sigma = \{$colorless, green, ideas, sleep, furiously$\}$
3. P

S $\rightarrow$ NP VP	NP $\rightarrow$ AdjP NP
VP $\rightarrow$ VP AdvP	
VP $\rightarrow$ V	NP $\rightarrow$ N_{pl}
AdvP $\rightarrow$ Adv	AdjP $\rightarrow$ Adj

Adj $\rightarrow$ colorless	Adj $\rightarrow$ green
N_{pl} $\rightarrow$ ideas	V $\rightarrow$ sleep
Adv $\rightarrow$ furiously	

4. S

We can derive the structure of a string.

- S$\Rightarrow$ NP VP
  $\Rightarrow$ N_{pl} VP
  $\Rightarrow$ ideas VP
  $\Rightarrow$ ideas V AdvP
  $\Rightarrow$ ideas sleep AdvP
  $\Rightarrow$ **ideas sleep Adv**

An example

1. $N = \{\text{S}, \text{NP}, \text{VP}, \text{AdjP}, \text{AdvP}\} \cup \{\text{N}_{pl}, \text{V}, \text{Adj}, \text{Adv}\}$
2. $\Sigma = \{$colorless, green, ideas, sleep, furiously$\}$
3. P

S $\rightarrow$ NP VP VP $\rightarrow$ VP AdvP VP $\rightarrow$ V AdvP $\rightarrow$ Adv	NP $\rightarrow$ AdjP NP NP $\rightarrow$ N_{pl} AdjP $\rightarrow$ Adj
Adj $\rightarrow$ colorless N_{pl} $\rightarrow$ ideas Adv $\rightarrow$ furiously	Adj $\rightarrow$ green V $\rightarrow$ sleep

4. S

We can derive the structure of a string.

- S$\Rightarrow$ NP VP
  $\Rightarrow$ N_{pl} VP
  $\Rightarrow$ ideas VP
  $\Rightarrow$ ideas V AdvP
  $\Rightarrow$ ideas sleep AdvP
  $\Rightarrow$ ideas sleep Adv
  $\Rightarrow$ **ideas sleep furiously**

An example

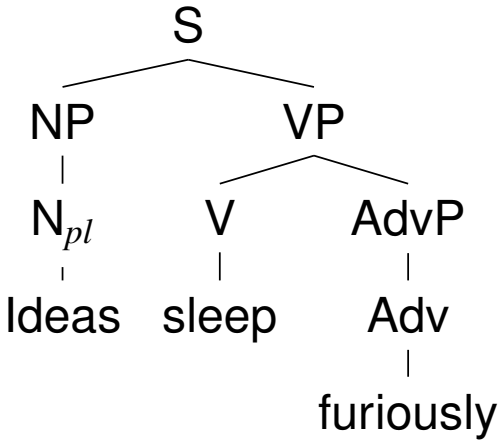
1. $N = \{\text{S}, \text{NP}, \text{VP}, \text{AdjP}, \text{AdvP}\} \cup \{\text{N}_{pl}, \text{V}, \text{Adj}, \text{Adv}\}$
2. $\Sigma = \{$colorless, green, ideas, sleep, furiously$\}$
3. P

S $\rightarrow$ NP VP VP $\rightarrow$ VP AdvP VP $\rightarrow$ V AdvP $\rightarrow$ Adv	NP $\rightarrow$ AdjP NP NP $\rightarrow$ N_{pl} AdjP $\rightarrow$ Adj
Adj $\rightarrow$ colorless N_{pl} $\rightarrow$ ideas Adv $\rightarrow$ furiously	Adj $\rightarrow$ green V $\rightarrow$ sleep

4. S

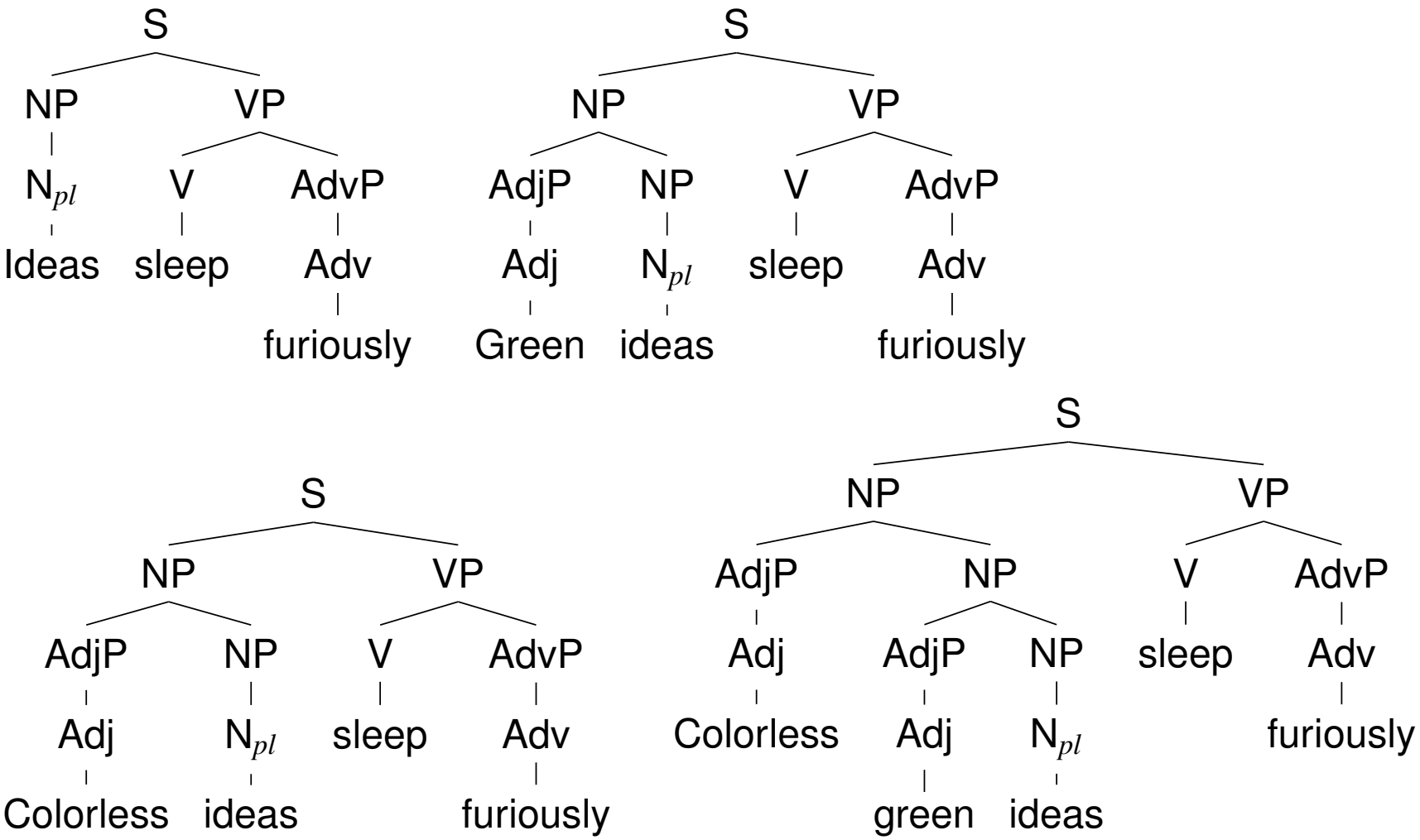
We can derive the structure of a string.

- S$\Rightarrow$ NP VP
  $\Rightarrow$ N_{pl} VP
  $\Rightarrow$ ideas VP
  $\Rightarrow$ ideas V AdvP
  $\Rightarrow$ ideas sleep AdvP
  $\Rightarrow$ ideas sleep Adv
  $\Rightarrow$ ideas sleep furiously



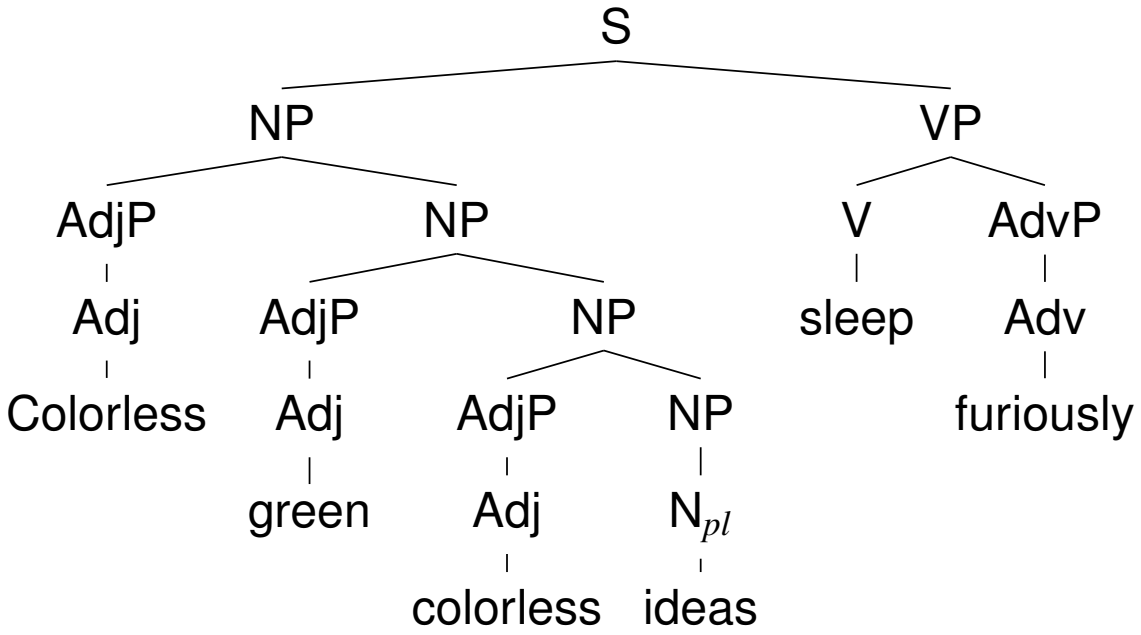
An example

We can define the language of a grammar by applying the productions.



Summary

- A context-free phrase-structure grammar provides a simple and mathematically precise mechanism for describing the methods by which phrases in some natural language are built from smaller blocks.
- The block structure of sentences is captured in a natural way.
- The basic recursive structure of sentences is described exactly.



Recursion

Recursion

the ability to place one component inside another component of the same type.

Recursion

Natural numbers

- $0 \leftarrow \emptyset$
- If n is a natural number, let $n + 1 \leftarrow n \cup \{n\}$

$$0 = \emptyset$$
$$1 = \{0\} = \{\emptyset\}$$
$$2 = \{0, 1\} = \{\emptyset, \{\emptyset\}\}$$
$$3 = \{0, 1, 2\} = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$$

Recursion

the ability to place one component inside another component of the same type.

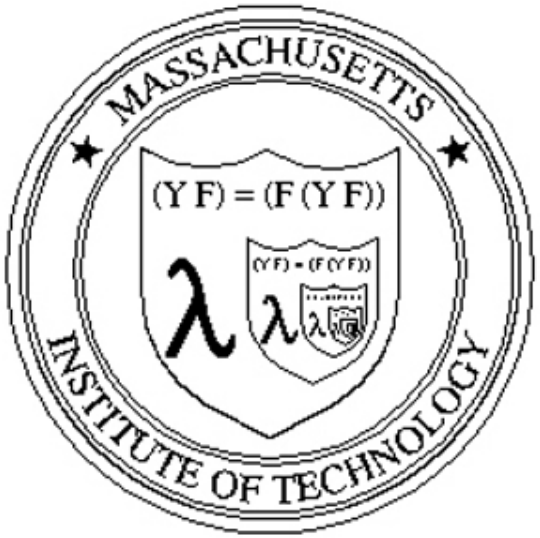
Recursion

GNU=GNU's Not Unix

Recursion

the ability to place one component inside another component of the same type.

Recursion



Recursion

the ability to place one component inside another component of the same type.

Recursion

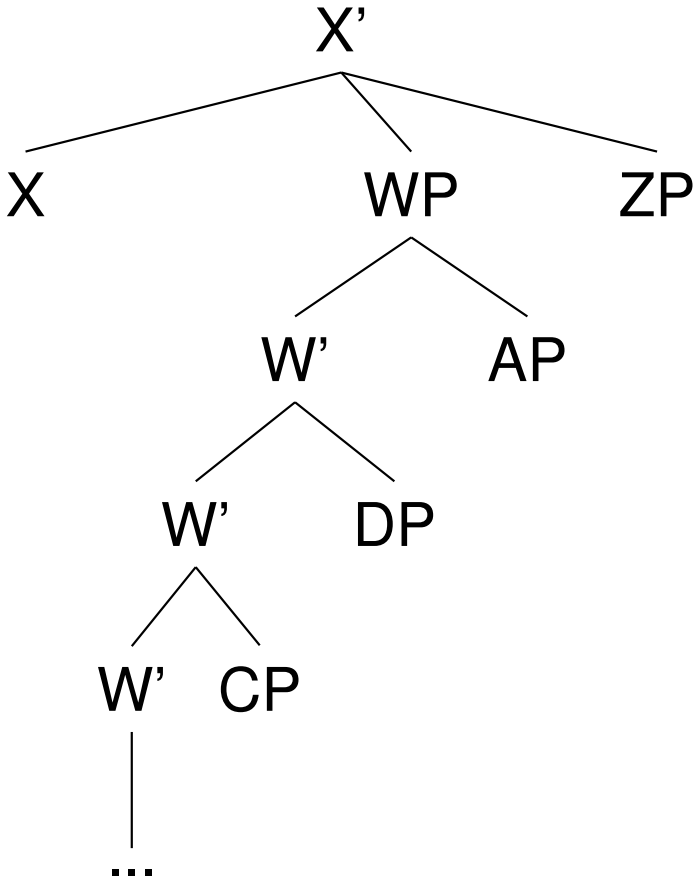
The narrow language faculty includes recursion and this is the only uniquely human component of the faculty of language.

Hauser

Recursion

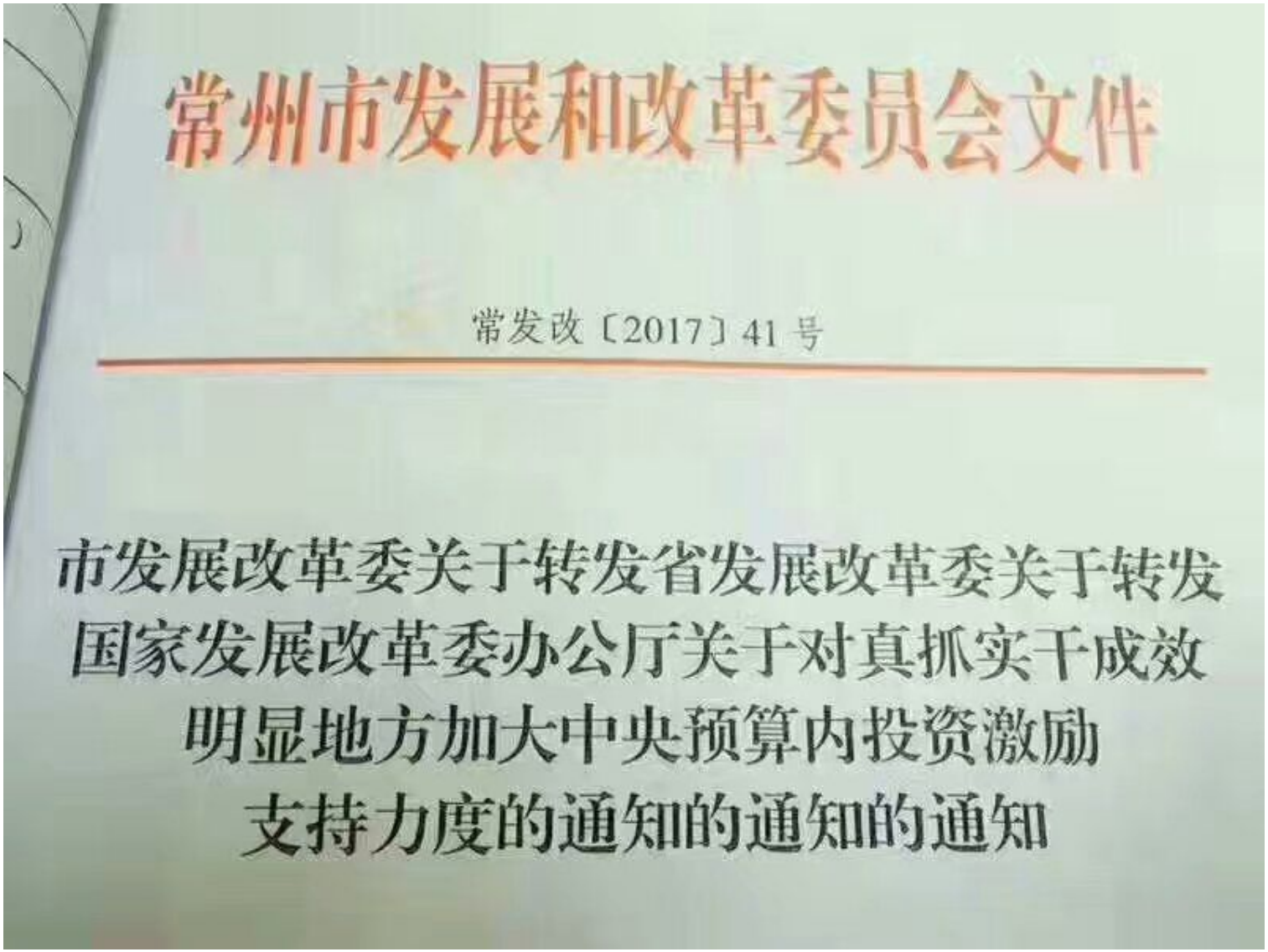
the ability to place one component inside another component of the same type.

Center embedding



Finite rules, infinite sentences!

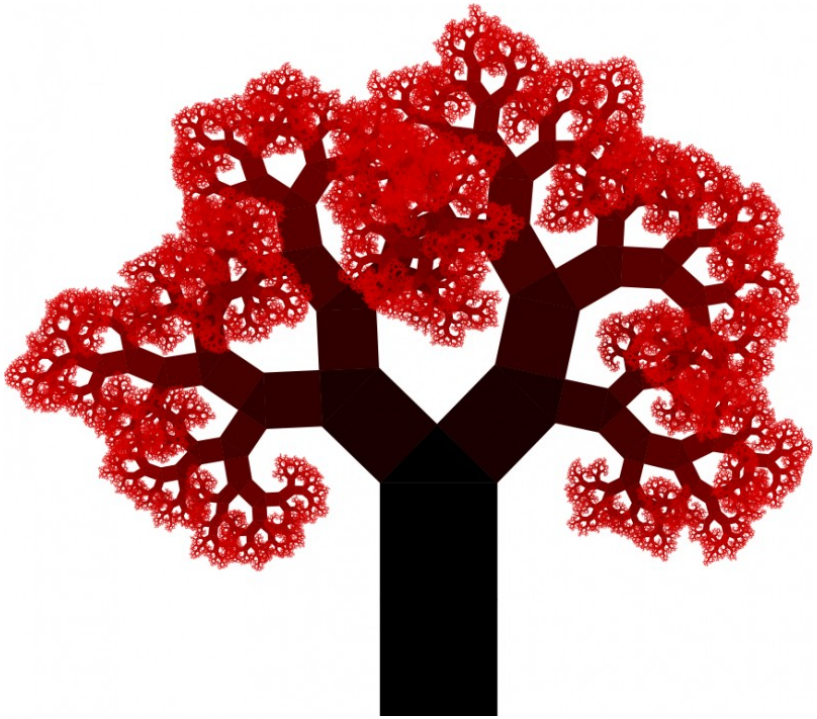
Center Embedding



Fun with context free

Context Free Art

www.contextfreeart.org/



```
startshape Tree [sat 1]

shape Tree {
  Trunk []
  Branch []
}

shape Trunk {
  SQUARE [y -0.5]
}

shape Branch {
  SQUARE [y +0.5]
  Fork [y (+1 + gap)]
}

shape Fork {
  Fork1 (cmin..sqrt(1 - cmin*cmin)) []
}

shape Fork1 (number c) {
  Triangle (c) []
  Branch [[x (c*c/2 - 0.5) y (c * sqrt(1 - c*c) / 2) r (90-asin(c)) y...
       +gap s c b 0.1]]
  Branch [[x (c*c/2) y (c * sqrt(1 - c*c) / 2) r (-asin(c)) y +gap s ...
      sqrt(1-c*c) b 0.1]]
}
```

Draw a tree all the time

Scientific theory

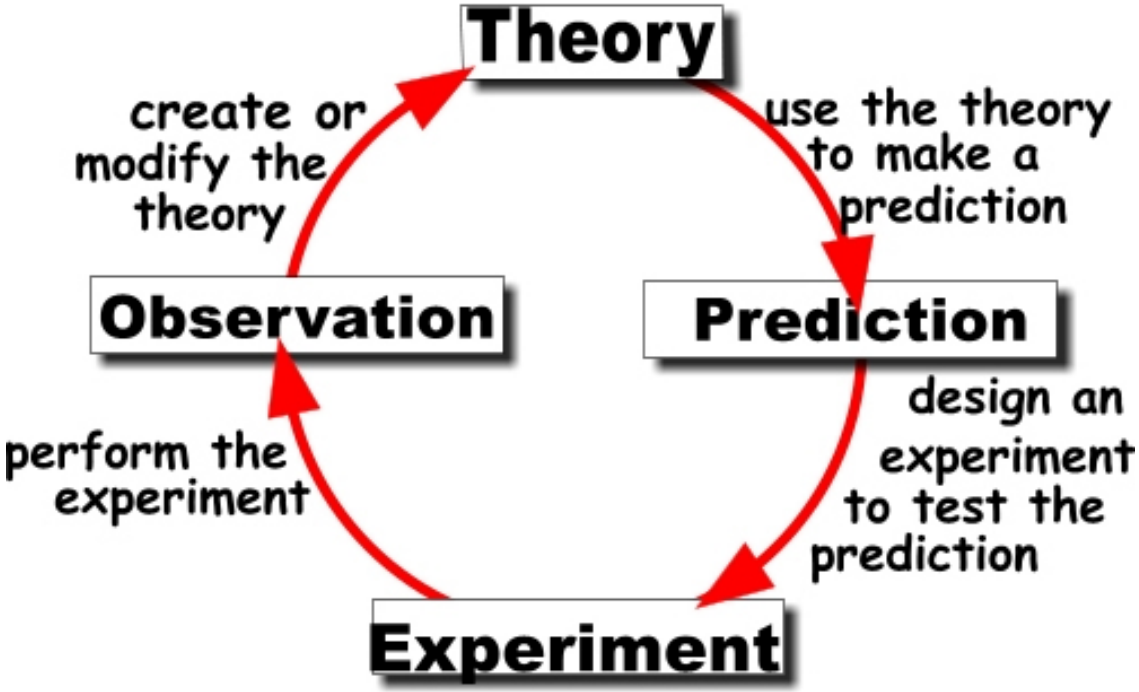
Wikipedia [en.wikipedia.org/wiki/Scientific_theory]

A scientific theory is a well-substantiated **explanation** of some aspect of the natural world that is acquired through the scientific method and repeatedly tested and confirmed through observation and experimentation.

A good scientific theory

- testable and make falsifiable predictions
- predictive power
- explanatory capability
- elegance and simplicity
- systematic

Scientific method



The expressiveness of CFG

Theorem

The copy language $\{ww|w \in \{a,b\}^*\}$ *is not context-free.*

Discontinuity

- A given word/phrase is separated from another word/phrase that it depends on.
- A direct connection cannot be established between the two words/phrases.

☹ CFG cannot handle discontinuities well.

Cross-serial dependencies

(1) a. that Charles lets Mary help Peter teach John to swim [English]

b. dass der Karl die Maria dem Peter den Hans schwimmen lehren helfen laesst [German]

c. dat Karel Marie Piet Jan laat helpen leren zwemmen [Dutch]

d. dass de Karl d'Maria em Peter de Hans laat hälfe lärne schwüme [Swiss German]

Cross-serial dependencies

(2) a. that Charles lets Mary help Peter teach John to swim [English]

b. dass der Karl die Maria dem Peter den Hans schwimmen lehren helfen laesst [German]

c. dat Karel Marie Piet Jan laat helpen leren zwemmen [Dutch]

d. dass de Karl d'Maria em Peter de Hans laat hälfe lärne schwüme [Swiss German]

Question

How to draw a tree?

Avoid misunderstanding

并提

(3) a. 耳目聪明

b. 自非亭午夜分,不见曦月

c. 千万条腿来千万只眼，也不够我走来也不够我看。

回文

(4) a. 东市买骏马，西市买鞍鞯，南市买辔头，北市买长鞭。

b. 秦时明月汉时关，万里长征人未还

c. 生的伟大，死的光荣

???

(5) a. 习近平、李克强、张德江、俞正声分别担任国家主席、国务院总理、人大委员长、政协主席。

b. 北京、华盛顿、伦敦、柏林、雅典分别是中国、美国、英国、德国、希腊的首都

Formal grammars

A grammar G consists of the following components:

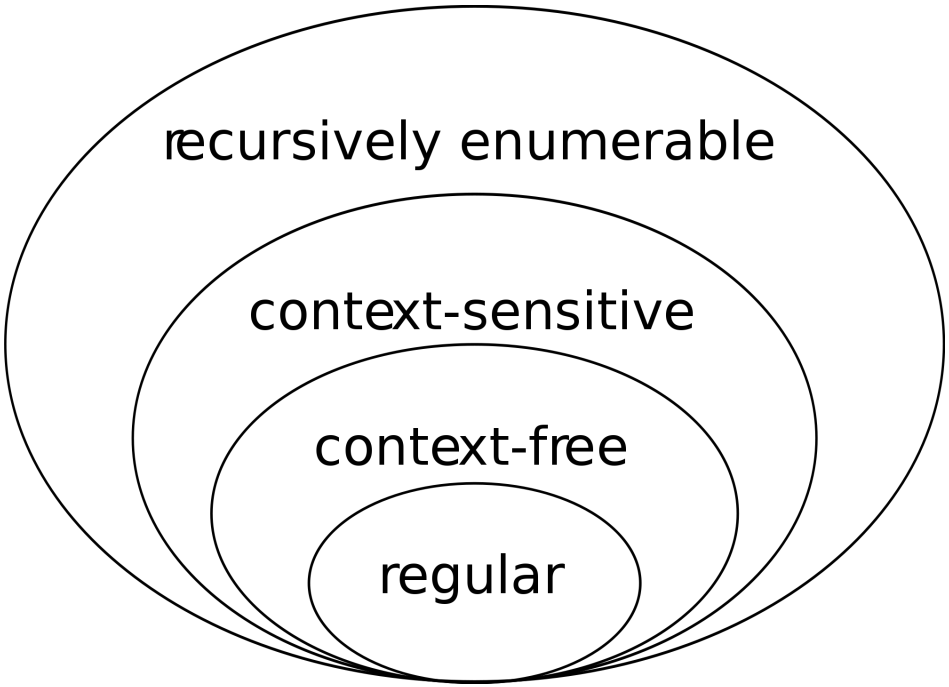
- A finite set Σ of terminal symbols that is disjoint from N.
- A finite set V of nonterminal symbols, none of which appear in strings formed from G.
- A distinguished nonterminal symbol that is the START symbol.
- A finite set P of production rules, each rule of the form

$$(\Sigma \cup N)^* N (\Sigma \cup N)^* \rightarrow (\Sigma \cup N)^*$$

  - * is the Kleene star operator, meaning repeating any times (including 0).
  - $\cup$ denotes set union
  - Each production rule maps from one string of symbols to another.
    - The left-hand side contains an arbitrary number of symbols and at least one of them is a nonterminal.

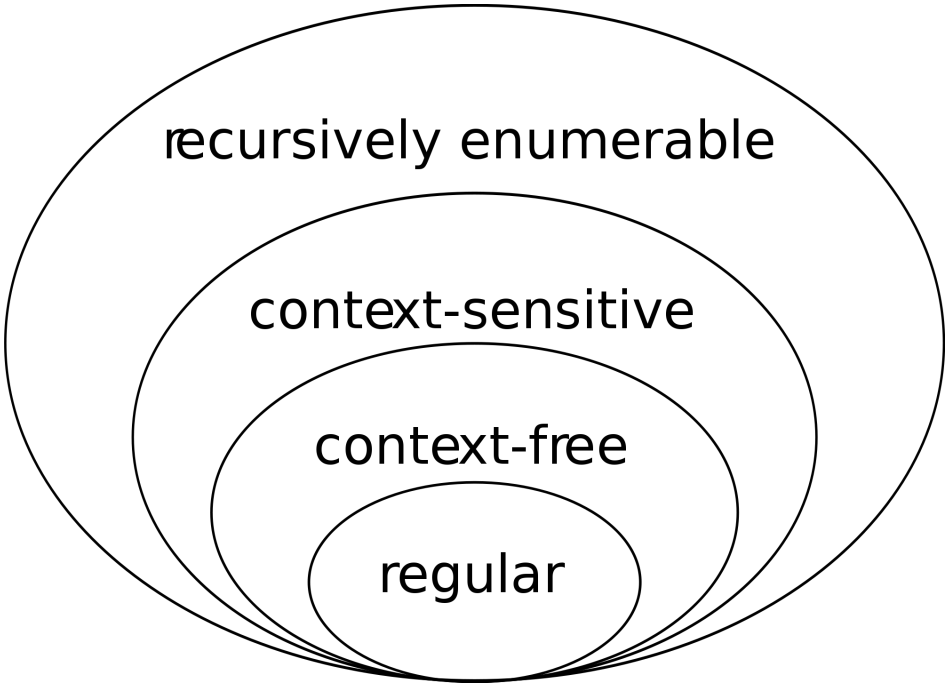
Chomsky Hierarchy

- Formal generative grammars can be classified into types now known as the Chomsky hierarchy.
- Different types of grammars have increasingly strict production rules and can express different formal languages.



Chomsky Hierarchy

Grammar	Languages	Production rules
Type-0	Recursively enumerable	$\alpha \rightarrow \gamma$
Type-1	Context-sensitive	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Type-2	Context-free	$A \rightarrow \gamma$
Type-3	Regular	$A \rightarrow a$
		$A \rightarrow aB$



A Type-0 Grammar for $\{ww|w \in \{a,b\}^*\}$

$$
\begin{aligned}
S &\to D_1 D_2 T \\
T &\to A_1 A_2 T \mid B_1 B_2 T \mid E \\
\alpha_2 \beta_1 &\to \beta_1 \alpha_2 \\
A_2 E &\to Ea \\
B_2 E &\to Eb \\
D_2 E &\to F \\
A_1 F &\to Fa \\
B_1 F &\to Fb \\
D_1 F &\to \varepsilon
\end{aligned}
$$

$\alpha, \beta \in \{A, B, D\}$.

- First create a starter mark D_1, then an end-mark of first word D_2, and then pairs of letters until the end E.
- The **meta-rule** sorts all A_1 before A_2, but without interchanging A_1 and B_1 or A_2 and B_2.
- The E stamp renders the A_2 and B_2 non-terminals and after reaching the middle it changes into F that renders A_1 and B_1.
- Finally, after F reaches the front, it disappears.

Comprehensive theory or model

Instead of a comprehensive theory, we devise a *model* that simplifies the phenomenon to capture some key aspect of it.

Daniel Everett states that Pirahã has no

- numbers
- grammatical recursion

Example

(6) a. Hand me the nails that Dan bought.

b. Give me the nails. Dan bought those very nails. They are the same.

Everett, D. L. (2008). *Don't sleep, there are snakes*.

Comprehensive theory or model

The Grammar of Happyniness

Reading

- Chap. 3. *Syntax: A Generative Introduction*.
- Chap. 1&2. *Syntactic: A Formal Introduction*.