Formal Languages and Linguistics

Pascal Amsili

Sorbonne Nouvelle, Lattice (CNRS/ENS-PSL/SN)

Cogmaster, september 2023

Overview

Example I

$$
\begin{array}{lcl}
S & \rightarrow & A\,B \\
A & \rightarrow & aA \\
 & | & b \\
B & \rightarrow & bBc \\
 & | & \varepsilon
\end{array}
$$

- Rewriting system
- Auxiliary vocabulary (N for non-terminal)
- Start symbol (engendered language)
- Multiple derivations
- Syntactic tree

$S \to AB$

$A \to aA$

$A \to b$

$B \to bBc$

$\quad | \; \varepsilon$

$A \to a^n b$

$B \to b^m c^m$

$S \to AB$

$S \to AB \overset{*}{\to} b$

$S \overset{*}{\to} ab$

$\alpha = \{b, ab, aab, \ldots$

$\alpha = \{a^n b b^m c^m \mid n, m \in \mathbb{N}\}$

Example II

$$
\begin{array}{rcl}
E & \rightarrow & E + E \\
 & | & E \times E \\
 & | & (\, E\,) \\
 & | & 0 \mid 1 \mid 2 \ldots 8 \mid 9
\end{array}
$$

- Syntactic ambiguity
- Semantic interpretation

$E \to E + E$

$E \to E \times E$

$E \to (E)$

$E \to 0 | 1 | 2 | 3 \ldots 9$

$\Sigma = \{0, 1, 2 \ldots, +, \times, (,)\}$

$N = \{E\}$

~~$2 \times \times 3$~~ ~~$(((3)))$~~

$2 \times 3 + 1$

$E \to E \times E \to E \times E + E \to 2 \times E + E \to 2 \times 3 + E \to 2 \times 3 + 1$

$E \to E \times E \to E \times E + E \to E \times E + 1 \to 2 \times E + 1 \to 2 \times 3 + 1$

<u>left-derivation</u>

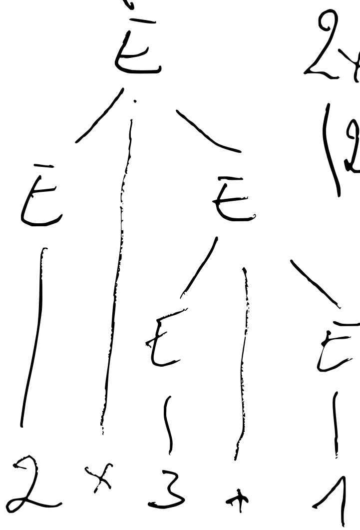
$E \to E \times E \to 2 \times E \to 2 \times E + E \to 2 \times 3 + E \to 2 \times 3 + 1$

$E \to E + E \to E \times E + E \to 2 \times E + E \to 2 \times 3 + E \to 2 \times 3 + 1$

$$E \to E \times E \to 2 \times E \to 2 \times E + E \to 2 \times 3 + E \to 2 \times 3 + 1$$

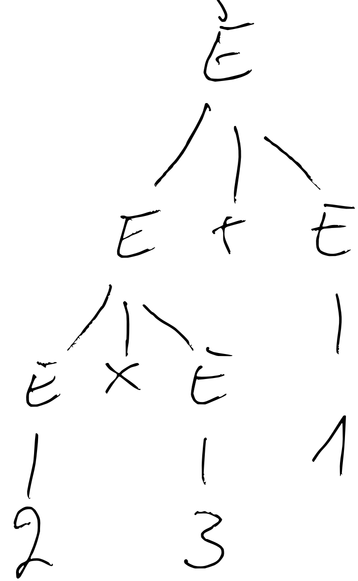
$$E \to E + E \to E \times E + E \to 2 \times E + E \to 2 \times 3 + E \to 2 \times 3 + 1$$

$$2 + 7 \times (3 + 1)$$



$$2 \times 3 + 1 = 7$$

$$(2 \times 3) + 1 = 7$$



$$E \to (E + E)$$

$$E \to (E \times E)$$

$$E \to 0 | 1 \dots 9$$

$$3 + 7 \times 2$$

E T F

$$E \to E + T \mid T$$

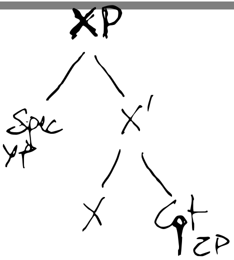
$$T \to T \times F \mid F$$

$$F \to (E)$$

$$\mid 0 | 1 \dots 9$$

X : N

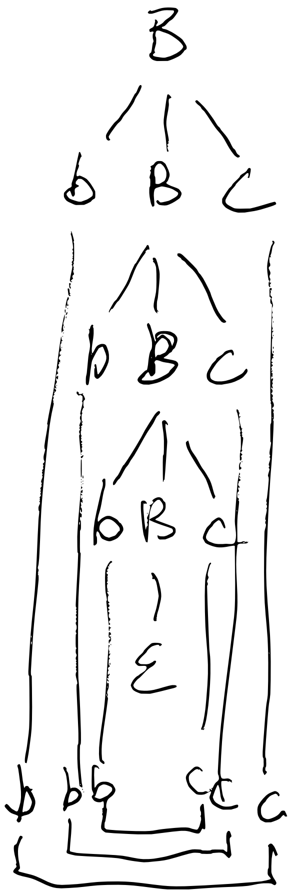
Example III



NP	$\rightarrow$	*Det N′*
N′	$\rightarrow$	*AdjP N′*
N′	$\rightarrow$	*N*
N′	$\rightarrow$	*N Cpt*
AdjP	$\rightarrow$	*Adj AdjP*
AdjP	$\rightarrow$	*Adj*
Cpt	$\rightarrow$	*P NP*
Det	$\rightarrow$	the | my
N	$\rightarrow$	cat | friend
Adj	$\rightarrow$	large | fierce
Prep	$\rightarrow$	of | to

- X-bar theory
- Recursive rules
- Center-embedding

the fierce friend of my cat


B
b B C
b B C
b B C
ε
b b b c c c

Overview

Formal grammar

$S \to aSb \mid \varepsilon$

$S \to S$

$BB \to AB$

$S \to aBc$

$aSa \to aBB$

~~$adc \to B$~~

Def. 12 ((Formal) Grammar)

A **formal grammar** is defined by $\langle \Sigma, N, S, P \rangle$ where

- Σ is an alphabet
- N is a disjoint alphabet (non-terminal vocabulary)
- $S \in N$ is a distinguished element of N, called the *axiom*
- P is a set of « *production rules* », namely a subset of the cartesian product $(\Sigma \cup N)^* N (\Sigma \cup N)^* \times (\Sigma \cup N)^*$.

(BB, AB)

Immediate Derivation

Def. 13 (Immediate derivation)

Let $\mathcal{G} = \langle \Sigma, N, S, P \rangle$ a grammar,
$r \in P$ a production rule, such that $r : A \longrightarrow u$ with $u \in (\Sigma \cup N)^*$;
$f, g \in (\Sigma \cup N)^*$ two "(proto-)words",

- f derives into g (immediate derivation) with the rule r (noted $f \xrightarrow{r} g$) iff
  $\exists v, w$ s.t. $f = vAw$ and $g = vuw$
- f derives into g (immediate derivation) in the grammar $\mathcal{G}$ (noted $f \xrightarrow{\mathcal{G}} g$) iff
  $\exists r \in P$ s.t. $f \xrightarrow{r} g$.

Sorbonne Nouvelle

Derivation

Def. 14 (Derivation)

$f \xrightarrow{\mathcal{G}*} g$ if $f = g$ or

$\exists f_0, f_1, f_2, ..., f_n$ s.t.

$f_0 = f$

$f_n = g$

$\forall i \in [1, n] : f_{i-1} \xrightarrow{\mathcal{G}} f_i$

Sorbonne Nouvelle

Engendered language

Def. 15 (Language engendered by a word)

Let $f \in (\Sigma \cup N)^*$.

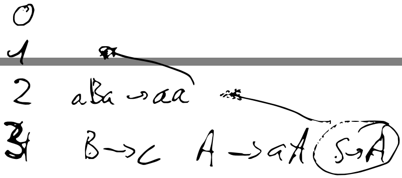
$L_{\mathcal{G}}(f) = \{g \in \Sigma^* / f \xrightarrow{\mathcal{G}*} g\}$

Def. 16 (Language engendered by a grammar)

The *language engendered by a grammar* $\mathcal{G}$ is the set of words of Σ^* derived from the axiom.

$L_{\mathcal{G}} = L_{\mathcal{G}}(S)$

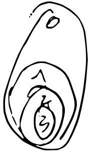
Overview



Principle

Define language families on the basis of properties of the grammars that generate them :

1. Four classes are defined, they are included one in another
2. A language is of type k if it **can** be recognized by a type k grammar (and thus, by definition, by a type $k-1$ grammar) ; and **cannot** be recognized by a grammar of type $k+1$.



Chomsky's hierarchy

Schützenberger

0 $aSb \to c$
1 $aSb \to acSb$
2 $S \to aSb$
3 $A \to aB$, $A \to b$, $A \to \varepsilon$

type 0 No restriction on $P \subset (X \cup V)^* V (X \cup V)^* \times (X \cup V)^*$.

type 1 (*context-sensitive* grammars) All rules of P are of the shape $(u_1 S u_2, u_1 m u_2)$, where u_1 and $u_2 \in (X \cup V)^*$, $S \in V$ and $m \in (X \cup V)^+$.

type 2 (*context-free* grammar) All rules of P are of the shape (S, m), where $S \in V$ and $m \in (X \cup V)^*$.

type 3 (*regular* grammars) All rules of P are of the shape (S, m), where $S \in V$ and $m \in X.V \cup X \cup \{\varepsilon\}$.

$u_1 S u_2 \to u_1 m u_2$ $A \to aA$

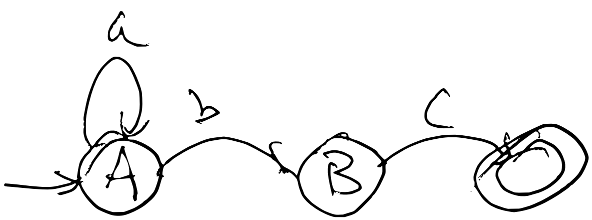
Sorbonne Nouvelle

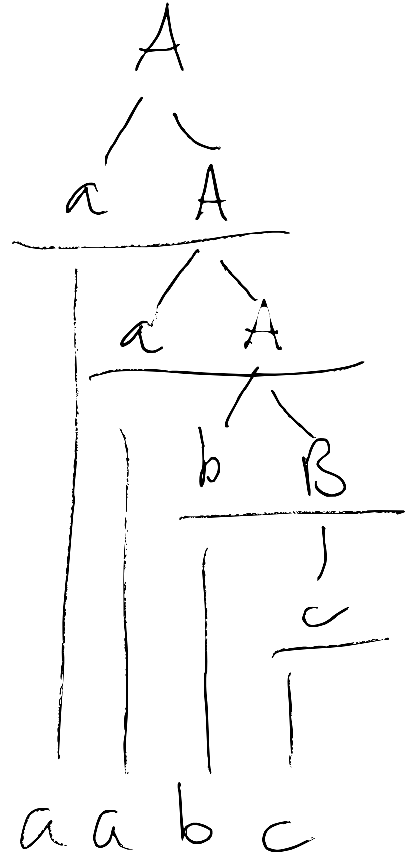
axiom: ~~A~~ S

$A \to aA$
$\quad | \; bB$

$S \to A$
$\quad | \; B$

$B \to c$





Examples

type 3:

$$\begin{array}{lcl} S & \rightarrow & aS \mid aB \mid bB \mid cA \\ B & \rightarrow & bB \mid b \\ A & \rightarrow & cS \mid bB \end{array}$$

Examples

type 3:

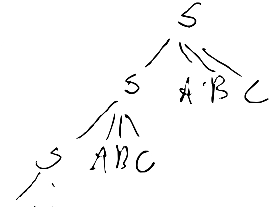
$$\begin{array}{lcl} S & \rightarrow & aS \mid aB \mid bB \mid cA \\ B & \rightarrow & bB \mid b \\ A & \rightarrow & cS \mid bB \end{array}$$

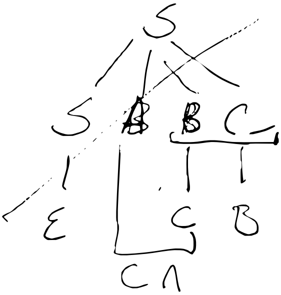
type 2:

$E \rightarrow E + T \mid T, T \rightarrow T \times F \mid F, F \rightarrow (E) \mid a$

Sorbonne Nouvelle

Example 1 type 0





Type 0:

$S \rightarrow SABC$	$AC \rightarrow CA$	$A \rightarrow a$
$S \rightarrow \varepsilon$	$CA \rightarrow AC$	$B \rightarrow b$
$AB \rightarrow BA$	$BC \rightarrow CB$	$C \rightarrow c$
$BA \rightarrow AB$	$CB \rightarrow BC$	

generated language :

$S \rightarrow SABC \rightarrow ABC \rightarrow BAC \rightarrow BCA \xrightarrow{*} bca$

Example 1 type 0

Type 0:

$$\begin{array}{lll} S \rightarrow SABC & AC \rightarrow CA & A \rightarrow a \\ S \rightarrow \varepsilon & CA \rightarrow AC & B \rightarrow b \\ AB \rightarrow BA & BC \rightarrow CB & C \rightarrow c \\ BA \rightarrow AB & CB \rightarrow BC & \end{array}$$

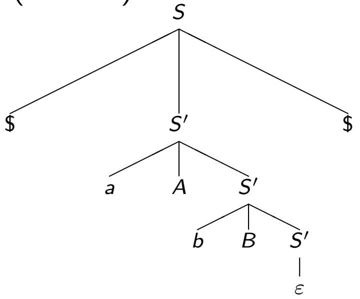
generated language : words with an equal number of a, b, and c.

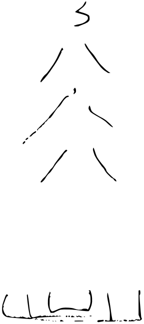
Example 2: type 0

Type 0:

$$
\begin{array}{llll}
S \to \$S'\$ & Aa \to aA & \$a \to a\$ \\
S' \to aAS' & Ab \to bA & \$b \to b\$ \\
S' \to bBS' & Ba \to aB & A\$ \to \$a \\
S' \to \varepsilon & Bb \to bB & B\$ \to \$b \\
 & & \$\$ \to \#
\end{array}
$$

Example 2: type 0 (cont'd)

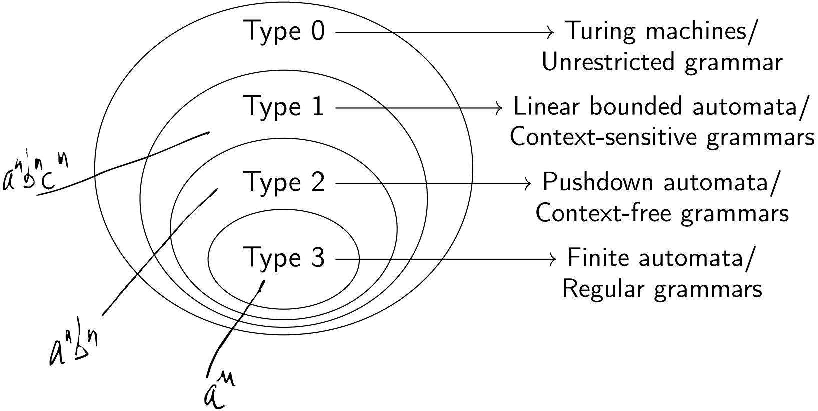




$	a	A	b	B	$
a	$	A	b	B	$
a	$	A	b	$	b
a	$	b	A	$	b
a	b	$	A	$	b
a	b	$	$	a	b
a	b	#		a	b

$S \rightarrow aSb \mid \varepsilon$

The Chomsky-Schützenberger hierarchy



Remarks

- Type 0 (Turing-recognizable) = recursively enumerable languages
  Type 1 (Turing-decidable) = recursive languages
- There are others ways to classify languages,
  - either on other properties of the grammars;
  - or on other properties of the languages
- Nested structures are preferred, but it's not necessary

The parsing problem: finding derivations

- Given a grammar G on some alphabet Σ...
- The **parsing problem** for G:

  Given some $w \in \Sigma$,
  what are the derivations (if any) of w in G?
- (Solving the parsing problem for G entails solving the recognition problem for $\mathcal{L}(G)$.)

Sorbonne Nouvelle

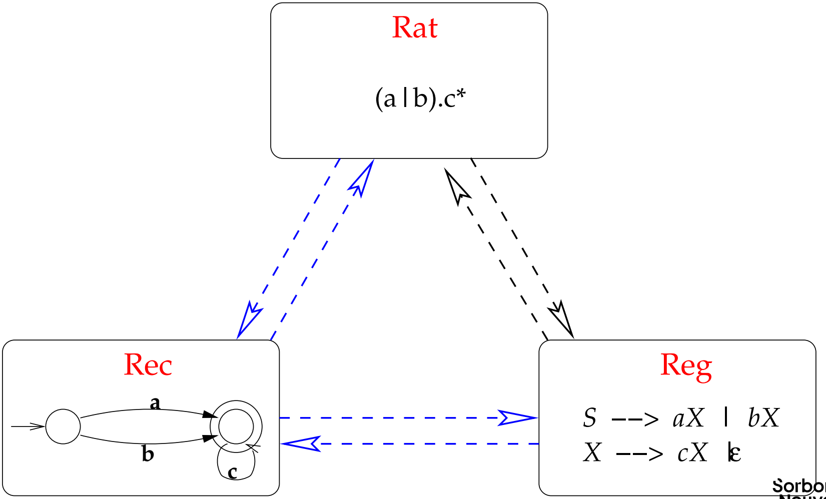
Syntactic complexity vs semantic expressivity

- Context-free grammars are commonly used to describe the syntax of many logical languages (PL, FOL), some programming languages, and parts of NL ($\rightarrow$ Day 2).
- Untyped λ-calculus: CF syntax, Turing-complete semantics. "How is this possible?"
- $\rightarrow$ The syntactic complexity and the semantic expressivity of interpreted languages are two distinct notions.
- Jot (https://en.wikipedia.org/wiki/Iota_and_Jot) is $\{0, 1\}$, a regular language, compositionally interpreted as a Turing-complete language.

The recognition/parsing problems are very general

- Consider any binary ("yes/no") problem P and see it as the set of inputs for which the answer is positive.
- Let *str* be a linearisation function for the possible inputs of P, and $L = \{str(in) \mid in \in P\}$.
- Solving P is equivalent to the recognition problem for L.
- More generally, any computable function f can be encoded as a grammar s.t. after parsing the input w, the output $f(w)$ can be read off the derivation.
- $\rightarrow$ One can compute "syntactically": a grammar is a program. (The parser is the machine that runs it.)
- The formalism of unrestricted grammars is a Turing-complete programming language. (syntactically regular?)

Back to regular languages



Sorbonne Nouvelle

Let's play with grammars

For each of the following grammars, give the generated language, and the type they have in Chomsky's hierarchy.

$$\begin{array}{lcl} S & \rightarrow & S_1 S_2 \\ S_1 & \rightarrow & aS_1 b \mid ab \\ S_2 & \rightarrow & cS_2 \mid c \end{array}$$

$$\begin{array}{lcl} S & \rightarrow & aSBC \\ S & \rightarrow & aBC \\ CB & \rightarrow & BC \\ aB & \rightarrow & ab \\ bB & \rightarrow & bb \\ bC & \rightarrow & bc \\ cC & \rightarrow & cc \end{array}$$

Let's play with grammars (cont'd)

Give a contex-free grammar that generates each of the following languages (alphabet $\Sigma = \{a, b, c\}$).

- $L_0 = \{w \in X^* \;/\; w = a^n \;;\; n \geq 0\}$
- $L'_0 = \{w \in X^* \;/\; w = a^n b^n ca \;;\; n \geq 0\}$
- $L_1 = \{w \in X^* \;/\; w = a^n b^n c^p ; n > 0 \text{ et } p > 0\}$
- $L_2 = \{w \in X^* \;/\; w = a^n b^n a^m b^m ; n, m \geq 1\}$
- $L'_3 = \{w \in X^* \;/\; |w|_a = |w]_b\}$
- $L_3 = \{w \in X^* \;/\; |w|_a = 2|w]_b\}$
- $L_4 = \{w \in X^* \;/\; \exists x \in X^* \text{ tq } w = x\overline{x}\}$
- $L_5 = \{w \in X^* \;/\; w = \overline{w}\}$

References I

Bar-Hillel, Yehoshua, Perles, Micha, & Shamir, Eliahu. 1961. On formal properties of simple phrase structure grammars. *STUF-Language Typology and Universals*, **14**(1-4), 143–172.

Chomsky, Noam. 1957. *Syntactic Structures*. Den Haag: Mouton & Co.

Chomsky, Noam. 1995. *The Minimalist Program*. Vol. 28. Cambridge, Mass.: MIT Press.

Gazdar, Gerald, & Pullum, Geoffrey K. 1985 (May). *Computationally Relevant Properties of Natural Languages and Their Grammars*. Tech. rept. Center for the Study of Language and Information, Leland Stanford Junior University.

Gibson, Edward, & Thomas, James. 1997. The Complexity of Nested Structures in English: Evidence for the Syntactic Prediction Locality Theory of Linguistic Complexity. *Unpublished manuscript, Massachusetts Institute of Technology*.

Joshi, Aravind K. 1985. *Tree Adjoining Grammars: How Much Context-Sensitivity is Required to Provide Reasonable Structural Descriptions?* Tech. rept. Department of Computer and Information Science, University of Pennsylvania.

Langendoen, D Terence, & Postal, Paul Martin. 1984. *The vastness of natural languages*. Basil Blackwell Oxford.

Mannell, Robert. 1999. *Infinite number of sentences*. part of a set of class notes on the Internet. `http://clas.mq.edu.au/speech/infinite_sentences/`.

Schieber, Stuart M. 1985. Evidence against the Context-Freeness of Natural Language. *Linguistics and Philosophy*, **8**(3), 333–343.

Stabler, Edward P. 2011. Computational perspectives on minimalism. *Oxford handbook of linguistic minimalism*, 617–643.

Steedman, Mark, *et al.* . 2012 (June). *Combinatory Categorial Grammars for Robust Natural Language Processing*. Slides for NASSLLI course `http://homepages.inf.ed.ac.uk/steedman/papers/ccg/nasslli12.pdf`.

Vijay-Shanker, K., & Weir, David J. 1994. The Equivalence of Four Extensions of Context–Free Grammars. *Mathematical Systems Theory*, **27**, 511–546.