CS_275 Automata and Formal Language Theory

Course Notes
Part II: The Recognition Problem (II)
Chapter II.1: The Chomsky Hierarchy (12.1)

Anton Setzer
(Based on a book draft by J. V. Tucker and K. Stephenson)
Dept. of Computer Science, Swansea University

http://www.cs.swan.ac.uk/∼csetzer/lectures/automataFormalLanguage/current/index.html

April 19, 2018

II.1.1 Administrative Issues

II.1.2 Overview Part II

II.1.3 Grammars and Derivations (10.2)
II.1.3.1. Derivations (10.2.3.)
II.1.3.2. Language Generation (10.2.4.)

II.1.4 The Chomsky Hierarchy (12.1)

II.1.1 Administrative Issues

II.1.2 Overview Part II

II.1.3 Grammars and Derivations (10.2)
II.1.3.1. Derivations (10.2.3.)
II.1.3.2. Language Generation (10.2.4.)

II.1.4 The Chomsky Hierarchy (12.1)

Disclaimer

- These notes are heavily based on
  - J. V. Tucker and K. Stephenson: *Data, Syntax and Semantics, Course Notes, Dept. of Computer Science, Swansea University, 2006. ([JS06])*

  Substantial parts are identical to that text.
- Numbers in brackets (e.g. **(10), (10.1)**) in section headings, definitions, etc. **refer to the sections in this book.**
- In general none of the material in these slides is considered as original material.

Administrative Issues

Lecturer of Part II and Part III:
Dr. A. Setzer
Dept. of Computer Science
University of Wales Swansea
Singleton Park
SA2 8PP
UK

Room: Room 952, Talbot Building
Tel.: (01792) 513368
Fax: (01792) 295651
Email: a.g.setzer@swansea.ac.uk
Home page: http://www.cs.swan.ac.uk/~csetzer/index.html

Home Page of the Module

- The homepage for the parts taught by Anton Setzer is located at http://www.cs.swan.ac.uk/~csetzer/lectures/automataFormalLanguage/current/index.html
- This page is linked from blackboard.
- All slides will be available on blackboard (via a link to those slides).
- Errors in the notes will be corrected on the slides continuously and noted on the **list of errata** available from the homepage and linked from blackboard.
- The homepage contains as well **additional material** for each section of the module.
  That material is not directly relevant for the exam.
- Assessment related material such as coursework, solutions, and the revision lecture will be made available on Blackboard.

Literature

[JS06] J. V. Tucker and K. Stephenson: *Data, Syntax and Semantics*. Course Notes, Dept. of Computer Science, Swansea University, 2006. Available from http://www.cs.swan.ac.uk/~csjvt/JVTTeaching/TPL.html

- **Main Course Text.**
  The slides used in this module will be heavily based on this text.

[HMU07] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman: *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 3rd Ed, 2006.

- **Main additional book** for this part.
- Classical text on theory of formal languages

Literature

- **Other Texts:**

[ASU88] Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, *Compilers. Principles, Techniques, and Tools.* 2nd Edition, Addison-Wesley 2006.

- The famous "Dragon Book" (because of the book cover of the first edition), the bible of compiler construction.

[Su05] T. A. Sudkamp, *Languages and Machines,* 3rd Edition, 2005.

[Lo97] K. C. Louden, *Compiler Construction: Principles and Practice,* Addison-Wesley, 1997.

II.1.1 Administrative Issues

II.1.2 Overview Part II

II.1.3 Grammars and Derivations (10.2)
II.1.3.1. Derivations (10.2.3.)
II.1.3.2. Language Generation (10.2.4.)

II.1.4 The Chomsky Hierarchy (12.1)

Overview over Part II: The Recognition Problem

A hierarchy of grammars.

II.1. The Chomsky Hierarchy.

Regular Languages:

II.2. Basics of Regular Languages and Expressions.
II.3. Finite State Automata.
II.4. Properties of Regular Languages.
Includes equivalence theorem and pumping lemma.

Context-free languages:

II.5. Properties of Context Free Grammars.
Includes the Pumping Lemma for context-free grammars.

Some Notations

- Let u, v be strings.
  - $|u|$ is the length of string u.
  - uv denotes the concatenation of strings u and v.
  - $u^n := \underbrace{u \cdots u}_{n \text{ times}}$, i.e. n repetitions of string u.

  (Applies to this module only).
- If A is a set of strings, then

$$\begin{aligned} A^* &= \{u_1 u_2 \cdots u_n \mid n \in \mathbb{N}, u_i \in A\} \\ A^+ &= \{u_1 u_2 \cdots u_n \mid n \in \mathbb{N}, n \geq 1, u_i \in A\} \end{aligned}$$

Especially we have for the empty word $\epsilon \in A^*$.

Some Notations

- "s.t." stand for "such that".
- x' is **not the derivative of** x but is a **name for a new variable**.
- $\llcorner\lrcorner$ stands for blank as a character (i.e. as a terminal symbol).
  In diagrams we sometimes write blank or Blank for it.
- On the slides defined items are denoted by $\sim$ and green colour.
  For instance "$\underset{\sim}{|u|}$ is the length of string u" defines the notation $|u|$.
- When repeating text from previous slides, we usually write it in orange colour.

Some Standard Mathematical Notations

- $\mathbb{N}$ is the set of natural numbers

$$\mathbb{N} = \{0, 1, 2, \ldots\}$$

Note that $0 \in \mathbb{N}$.

- For sets A, B

$$A \times B := \{(a, b) \mid a \in A \wedge b \in B\}$$

for instance if

$$\begin{array}{rcl} A & = & \{0, 1\} \\ B & = & \{3, 4, 5\} \end{array}$$

then

$$A \times B = \{(0, 3), (0, 4), (0, 5), (1, 3), (1, 4), (1, 5)\}$$

Some Standard Mathematical Notations

- For a set A we have

$$A^n := \underbrace{A \times \cdots \times A}_{n \text{ times}}$$

especially

$$A^2 = A \times A$$

For instance

$$\mathbb{N}^2 = \mathbb{N} \times \mathbb{N}$$

Some Standard Mathematical Notations

- A function

$$f : A \times B \to C$$

is a function which has as arguments $a \in A$ and $b \in B$ and returns an element in C.
So we write

$f(a, b) =$ any expression using a and b which gives an element in C

For instance we write

$$f(x, y) = (x + y * 2)^x$$

Note (common mistake) we

don't write $f = (x + y * 2)^x$

Some Standard Mathematical Notations

- $f : \mathbb{N}^2 \to \mathbb{N}$ means that
f is a function taking two natural numbers and returning one natural number.
Note that we write for a function

$$f(n, m) = \cdots$$

for instance

$$f(n, m) = n + 2 * m$$

- Often we define a function by case distinction e.g.

$$f : \mathbb{N} \to \mathbb{N}$$
$$f(n) = \begin{cases} 3 * n & \text{if } n < 7 \\ n + 5 & \text{otherwise} \end{cases}$$

Excellent Links

- Wikipedia page on Cartesian Product ($A \times B$): http://en.wikipedia.org/wiki/Cartesian_product
- Wikipedia page on notations for functions: http://en.wikipedia.org/wiki/Function_%28mathematics%29#Notation

Notations w^{R}, L^{R}

- If w is a word, w^{R} is the result of reversing the word. For instance, if $w = abc$, then $w^{\mathrm{R}} = cba$.
- If L is a language,

$$L^{\mathrm{R}} := \{w^{\mathrm{R}} \mid w \in L\}$$

For instance if

$$L = \{abc, def, ghi\}$$

then

$$L^{\mathrm{R}} = \{cba, fed, ihg\}$$

Use of Colour

- **Note** Some of the slides contain some coloured parts. The colours are indistinguishable in the black and white handouts. It is recommended to look at them using the online version (on blackboard and on the web page).

Availability of Material of this Lecture

- Spare copies of the notes will be put into the tray for this module will be available on the ground floor of the Talbot building near PC lab 043.
- There is a link on blackboard to the lecture notes (book) of John Tucker. ([JS06])
  - For this part of the module, no notes from John Tucker's book will be photocopied, but if you have problems understanding something you can study his notes for clarification.
  - The slides will follow very closely John Tucker's book.
  - Note that numbers such as (12.1) indicate to which sections of John Tucker's book the current slides correspond.
- There is a list of errata available

Webpage

- A link to the web page is at the front page of each set of slides.
- There is as well a link on blackboard.
- The slides will be continually updated after each lecture.
  So if you have problems with some material, it might be worth to check whether the online version has been updated.

II.1.1 Administrative Issues

II.1.2 Overview Part II

II.1.3 Grammars and Derivations (10.2)
II.1.3.1. Derivations (10.2.3.)
II.1.3.2. Language Generation (10.2.4.)

II.1.4 The Chomsky Hierarchy (12.1)

From BNF to General Grammars

You have seen grammars in BNF such as

bnf	*Letter*
rules	
⟨Letter⟩	::= ⟨LowerCase⟩ | ⟨UpperCase⟩
⟨LowerCase⟩	::= **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **j** | **k** | **l** | **m** | **n** | **o** | **p** | **q** | **r** | **s** | **t** | **u** | **v** | **w** | **x** | **y** | **z**
⟨UpperCase⟩	::= **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** | **M** | **N** | **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** | **W** | **X** | **Y** | **Z**

Generalisation

- **BNF** is a specific user friendly notation for context free grammars (see II.1.2) which are grammars which are simple structure.
- It is mainly used for describing the syntax of programming languages.
- General grammar notation which allows to define the more complex grammars of the Chomsky Hierarchy are denoted traditionally using the following conventions:
  - Nonterminals are **not** written in angle brackets such as ⟨UpperCase⟩.
  - Terminals are **not** written in boldface.
  - We write $\longrightarrow$ instead of ::=.
  - We usually don't use "|" for writing multiple clauses, and instead write one clause each.
    However sometimes one uses it as a short hand.

Example 1: Letter

grammar	Letter		
terminals	a,b,c,. . .,z,A,B,. . .,Z,␣		
nonterminals	Letter, LowerCase, UpperCase		
start symbol	Letter		
productions	Letter	$\longrightarrow$	LowerCase
	Letter	$\longrightarrow$	UpperCase
	LowerCase	$\longrightarrow$	a
	LowerCase	$\longrightarrow$	b
		. . .	
	LowerCase	$\longrightarrow$	z

Example 1: Letter (Continued)

UpperCase	$\longrightarrow$	A
UpperCase	$\longrightarrow$	B
	. . .	
UpperCase	$\longrightarrow$	Z

Definition of Grammars (10.2.1.)

Definition

A **Grammar** $G = (T, N, S, P)$ consists of

1. a finite set T called the **alphabet** or set of **terminal symbols**,
2. a finite set N of **non-terminal symbols** or **variable symbols** such that $T \cap N = \emptyset$,
3. a special non-terminal symbol $S \in N$ called the **start symbol**,
4. a finite set P of substitution or rewrite rules, called **productions**, each of which has the form $u \longrightarrow v$ where
   - 4.1 The left hand string $u \in (T \cup N)^+$ (esp. u is non-empty),
   - 4.2 The right hand string $v \in (T \cup N)^*$.

Presentation of Grammars (Reminder)

We present a grammar as a 4-tuple $G = (T, N, S, P)$ and also use a displayed version:

grammar	G
terminals	T
nonterminals	N
start symbol	S
productions	P

Alphabet, Terminal, Token

- Elements of the **alphabet** are **terminals**.
- In compiler technology, strings as sequences of characters are often first broken down into **tokens**.
  - A keyword such as in Java "class" would be a token. Each character of an identifier is a token.
- Tokens then serve in the next phase of the compiler, the parsing of a usually context-free grammar (see later) as elements of the alphabet.
- **Non-terminals** are intermediate symbols used in a derivation, which starts with the **start-symbol** (which is as well a non-terminal).
  - When we have derived a word of the alphabet all non-terminals have been reduced to strings of the alphabet.

Example 2: Grammar for English Sentence

grammar	SimpleEnglishGrammarExample		
terminals	a,b,c,...,z,A,B,...,Z,␣		
nonterminals	Sentence, Nounphrase, Verbphrase,Verb		
start symbol	Sentence		
productions	Sentence	$\longrightarrow$	Nounphrase ␣ Verbphrase
	Verbphrase	$\longrightarrow$	Verb ␣ Nounphrase
	Verb	$\longrightarrow$	g o
	Nounphrase	$\longrightarrow$	I
	Nounphrase	$\longrightarrow$	h o m e

Startsymbol, Terminal, Nonterminal

Sentence	$\longrightarrow$	Nounphrase ␣ Verbphrase
Verbphrase	$\longrightarrow$	Verb ␣ Nounphrase
Verb	$\longrightarrow$	g o
Nounphrase	$\longrightarrow$	I
Nounphrase	$\longrightarrow$	h o m e

In the above example we have

- "Sentence" is the **start symbol**.
- "Sentence", "Nounphrase", "Verbphrase" are **non-terminals**.
- "I", "g", "o" are **terminals**.

Derivation steps (Informal; see 10.2.3)

- Let G be a grammar.
- We write

  $s \Rightarrow t$ "t derives in one step from s"

  if there exists a production $v \longrightarrow w$ such that
  s contains a substring v
  and t is obtained by replacing in s one such occurrence of v by w.
- A derivation from v to w is a sequence of derivation steps

$$v = v_0 \Rightarrow v_1 \Rightarrow \cdots \Rightarrow v_n = w$$

Example

Sentence ⟶ Nounphrase ␣ Verbphrase
Verbphrase ⟶ Verb ␣ Nounphrase
Verb ⟶ g o
Nounphrase ⟶ I
Nounphrase ⟶ h o m e

In the above example we have

NounPhrase ␣ VerbPhrase
⇒ I ␣ VerbPhrase
⇒ I ␣ Verb ␣ NounPhrase
⇒ I ␣ g o ␣ NounPhrase
⇒ I ␣ g o ␣ h o m e

is a derivation of
"I ␣ g o ␣ h o m e" from "NounPhrase ␣ VerbPhrase".

Derivations (Informal; see 10.2.3)

A grammar G defines a formal language $L(G) \subseteq T^*$.
The elements of $L(G)$ are the set of strings consisting of **non-terminals** only, we can can derive in G from the start symbol S.
We will first give some examples and then a formal definition of $L(G)$.

Example 2: English Grammar

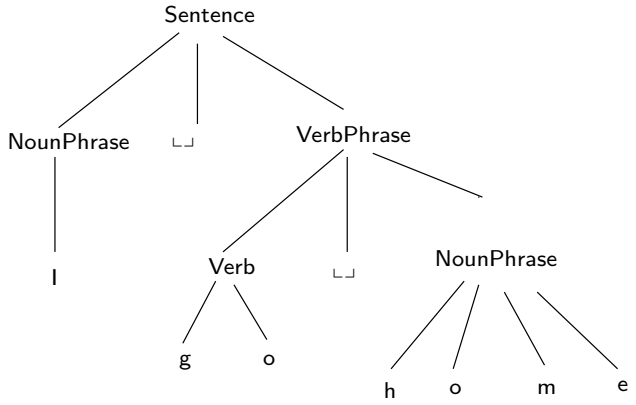
Start symbol sentence
Sentence ⟶ Nounphrase ␣ Verbphrase
Verbphrase ⟶ Verb ␣ Nounphrase
Verb ⟶ g o
Nounphrase ⟶ I
Nounphrase ⟶ h o m e

The following derives the sentence "I go home":

Sentence ⇒ NounPhrase ␣ VerbPhrase
⇒ I ␣ VerbPhrase
⇒ I ␣ Verb ␣ NounPhrase
⇒ I ␣ g o ␣ NounPhrase
⇒ I ␣ g o ␣ h o m e

Parse Tree for an English Sentence

The derivation above corresponds to the following parse tree:



Parse Trees vs Derivations

- Derivations as above are more general than parse trees.
- They can be used for all types of the Chomsky Hierarchy (introduced below).
- Parse tree can be used only for regular and context free grammars.

Eight different Derivations of "I go home"

There are eight derivations of I go home:

- Sentence ⇒ NounPhrase ␣ VerbPhrase
  ⇒ I ␣ VerbPhrase
  ⇒ I ␣ Verb ␣ NounPhrase
  ⇒ I ␣ g o ␣ NounPhrase
  ⇒ I ␣ g o ␣ h o m e
- Sentence ⇒ NounPhrase ␣ VerbPhrase
  ⇒ I ␣ VerbPhrase
  ⇒ I ␣ Verb ␣ NounPhrase
  ⇒ I ␣ Verb ␣ h o m e
  ⇒ I ␣ g o ␣ h o m e

Eight different Derivations of "I go home"

- Sentence ⇒ NounPhrase ␣ VerbPhrase
  ⇒ NounPhrase ␣ Verb ␣ NounPhrase
  ⇒ I ␣ Verb ␣ NounPhrase
  ⇒ I ␣ g o ␣ NounPhrase
  ⇒ I ␣ g o ␣ h o m e
- Sentence ⇒ NounPhrase ␣ VerbPhrase
  ⇒ NounPhrase ␣ Verb ␣ NounPhrase
  ⇒ I ␣ Verb ␣ NounPhrase
  ⇒ I ␣ Verb ␣ h o m e
  ⇒ I ␣ g o ␣ h o m e

Eight different Derivations of "I go home"

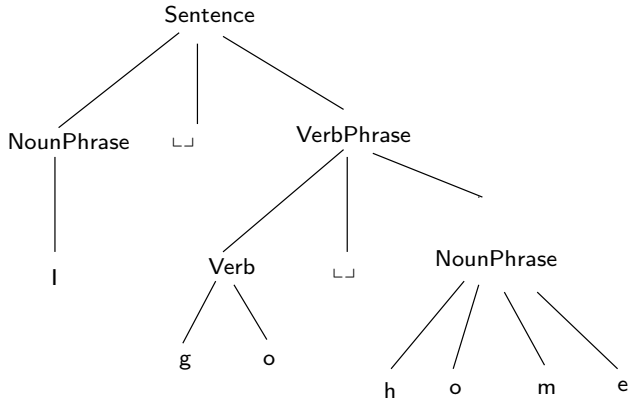
- Sentence ⇒ NounPhrase ␣ VerbPhrase
  ⇒ NounPhrase ␣ Verb ␣ NounPhrase
  ⇒ NounPhrase ␣ g o ␣ NounPhrase
  ⇒ I ␣ g o ␣ NounPhrase
  ⇒ I ␣ g o ␣ h o m e
- Sentence ⇒ NounPhrase ␣ VerbPhrase
  ⇒ NounPhrase ␣ Verb ␣ NounPhrase
  ⇒ NounPhrase ␣ g o ␣ NounPhrase
  ⇒ NounPhrase ␣ g o ␣ h o m e
  ⇒ I ␣ g o ␣ h o m e

Eight different Derivations of "I go home"

- Sentence $\Rightarrow$ NounPhrase ␣ VerbPhrase
  $\Rightarrow$ NounPhrase ␣ Verb ␣ NounPhrase
  $\Rightarrow$ NounPhrase ␣ Verb ␣ h o m e
  $\Rightarrow$ I ␣ Verb ␣ h o m e
  $\Rightarrow$ I ␣ g o ␣ h o m e
- Sentence $\Rightarrow$ NounPhrase ␣ VerbPhrase
  $\Rightarrow$ NounPhrase ␣ Verb ␣ NounPhrase
  $\Rightarrow$ NounPhrase ␣ Verb ␣ h o m e
  $\Rightarrow$ NounPhrase ␣ g o ␣ h o m e
  $\Rightarrow$ I ␣ g o ␣ h o m e

Parse Tree for an English Sentence

All the above 8 derivations correspond to the same parse tree



Example 3

Consider the grammar

$$G^{[01]^*1} = (\{0,1\}, \{S\}, S, \{S \longrightarrow 1, S \longrightarrow 0S, S \longrightarrow 1S\})$$

displayed as follows

grammar	$G^{[01]^*1}$
terminals	$0, 1$
nonterminals	S
start symbol	S
productions	$S \longrightarrow 1, S \longrightarrow 0S, S \longrightarrow 1S$

Example 3 (Generation of Strings)

We assign numbers to the production rules:
Rule 1 $S \longrightarrow 1$
Rule 2 $S \longrightarrow 0S$
Rule 3 $S \longrightarrow 1S$
A derivation of $1001 \in L(G^{[01]^*1})$ is as follows:

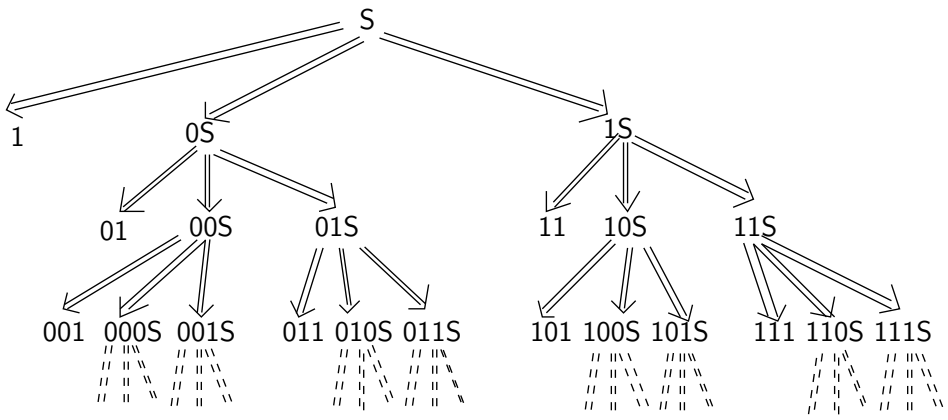
$$\begin{array}{rcll} S & \Rightarrow & 1S & \text{Rule 3} \\ & \Rightarrow & 10S & \text{Rule 2} \\ & \Rightarrow & 100S & \text{Rule 2} \\ & \Rightarrow & 1001 & \text{Rule 1} \end{array}$$

A derivation of $011 \in L(G^{[01]^*1})$ is as follows:

$$\begin{array}{rcll} S & \Rightarrow & 0S & \text{Rule 2} \\ & \Rightarrow & 01S & \text{Rule 3} \\ & \Rightarrow & 011 & \text{Rule 1} \end{array}$$

Derivations of Strings in $G^{\{01\}^*1}$

The following shows all strings derivable in this grammar.
Note that this is **not** a derivation tree, its the tree of all possible derivations in this language.



$L(G^{\{01\}^*1})$

We can see that the elements of $L(G^{\{01\}^*1})$ are the strings in $\{0,1\}^*$ which end with a 1:

$$L(G^{\{01\}^*1}) = \{w1 \mid w \in \{0,1\}^*\}$$

Remark

$G^{\{01\}^*1}$ is an example of a regular grammar (this notion will be introduced in II.1.3).

Example 4 (Grammars)

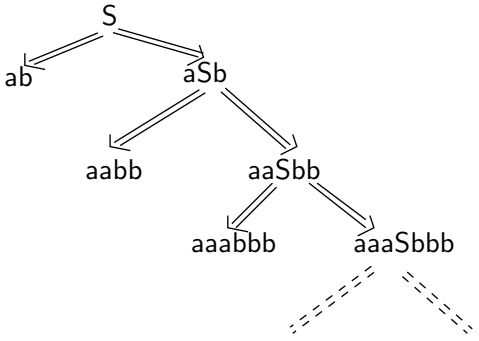
Consider the grammar

$$G^{a^n b^n} = (\{a,b\},\{S\},S,\{S \longrightarrow ab, S \longrightarrow aSb)$$

displayed as follows

grammar	$G^{a^n b^n}$
terminals	a, b
nonterminals	S
start symbol	S
productions	$S \longrightarrow ab, S \longrightarrow aSb$

Derivations of strings in $G^{a^n b^n}$



Note again that this is **not** a derivation tree, but a tree determining all possible derivable strings in this language.

$L(G^{a^n b^n})$

We can see that the elements of $L(G^{a^n b^n})$ are the strings of the form $a^n b^n$:

$$L(G^{a^n b^n}) = \{a^n b^n \mid n \geq 1\}$$

Remark

$G^{a^n b^n}$ is an example of a context-free grammar (this notion will be introduced later).

Example 5 (Grammars)

Consider the grammar

$$G^{a^n b^n c^n} = (\{a, b, c\}, \{S, B, C\}, S, \{S \longrightarrow aSBC, S \longrightarrow aBC, CB \longrightarrow BC, aB \longrightarrow ab, bB \longrightarrow bb, bC \longrightarrow bc, cC \longrightarrow cc\})$$

displayed as follows

grammar	$G^{a^n b^n c^n}$
terminals	a, b, c
nonterminals	S, B, C
start symbol	S
productions	$S \longrightarrow aSBC, S \longrightarrow aBC,$ $CB \longrightarrow BC,$ $aB \longrightarrow ab, bB \longrightarrow bb, bC \longrightarrow bc, cC \longrightarrow cc.$

Example Derivation

A derivation of *aaabbbccc* in Example 5 is as follows:

$$\begin{aligned}
\mathbf{S} &\Rightarrow aSBC &&= a\mathbf{S}BC \\
&\Rightarrow aaSBCBC &&= aa\mathbf{S}BCBC \\
&\Rightarrow aaaBCBCBC &&= aaaBCB\mathbf{CB}C \\
&\Rightarrow aaaBCBBCC &&= aaaB\mathbf{CB}BCC \\
&\Rightarrow aaaBBCBCC &&= aaaBB\mathbf{CB}CC \\
&\Rightarrow aaaBBBCCC &&= aa\mathbf{aB}BBCCC \\
&\Rightarrow aaabBBCCC &&= aaa\mathbf{bB}BCCC \\
&\Rightarrow aaabbBCCC &&= aaabb\mathbf{B}CCC \\
&\Rightarrow aaabbbCCC &&= aaabbb\mathbf{bC}CC \\
&\Rightarrow aaabbbcCC &&= aaabbbc\mathbf{cC}C \\
&\Rightarrow aaabbbccC &&= aaabbbcc\mathbf{C} \\
&\Rightarrow aaabbbccc
\end{aligned}$$

(**Blue parts** denote the left hand and red parts the right hand side of the production applied)

$L(G^{a^n b^n c^n})$

We can see that the elements of $L(G^{a^n b^n c^n})$ are the strings of the form $a^n b^n c^n$:

$$L(G^{a^n b^n c^n}) = \{a^n b^n c^n \mid n \geq 1\}$$

Remark

- *$G^{a^n b^n c^n}$ is often in books and web pages presented as an example of a context-sensitive grammar (this notion will be introduced later).*
- *It is not, but can be transformed into a context sensitive one, see Sect. II.1.3, Example 5, below.*

II.1.3 Grammars and Derivations (10.2)
II.1.3.1. Derivations (10.2.3.)
II.1.3.2. Language Generation (10.2.4.)

Derivations (10.2.3.)

Definition

1. Let $G = (T, N, S, P)$ be a grammar, $w \in (T \cup N)^+$ be a non-empty word, $w' \in (T \cup N)^*$ be a possibly empty word.

   We say that **w' is derived from w in one step**, with notation

   $$w \Rightarrow_G w'$$

   iff there exists a production $u \longrightarrow v \in P$ such that
   - u occurs in w,
   - w' is the result of replacing one occurrence of u in w by v.

Definition $\Rightarrow$ (Cont.)

Definition (Cont)

2. *We say as well* **w' is immediately generated from w** *or* **w' is one-step generated from w**, *or short*

   $$w \Rightarrow w'$$

   for $w \Rightarrow_G w'$.

- So w has the form sut for some $s, t \in (T \cup N)^*$, and $w' = svt$ where $u \longrightarrow v$ is a production.
- Informally w' is the result of replacing in w a substring, which is equal to the right hand string of a production by the left hand string of that production.

Definition $\Rightarrow^*$ (Cont.)

Definition

3. Let $G = (T, N, S, P)$ be a grammar, $w, w' \in (T \cup N)^*$ be a possibly empty words.

   We say that **w' is derived from w**, with notation $w \Rightarrow_G^* w'$ iff

   - either $w = w'$
   - or there exists a sequence of words $w_0, \ldots, w_n \in (T \cup N)^*$ s.t.
     - $w_0 = w$,
     - $w_n = w'$
     - for $1 \leq i \leq n-1$ we have $w_i \Rightarrow_G w_{i+1}$.

So $w \Rightarrow_G^* w'$ iff there exists $w_0, \ldots, w_n$ s.t.

$$w = w_0 \Rightarrow_G w_1 \Rightarrow_G w_2 \Rightarrow_G \cdots \Rightarrow_G w_n = w'$$

where $n = 1$ is allowed, which means $w \Rightarrow_G^* w$ always holds

Definition $\Rightarrow^*$ (Cont.)

Definition (Cont)

4. *We say as well* w' **is generated from** w *or*

$$w \Rightarrow^* w'$$

for $w \Rightarrow^*_G w'$.

Example

In example 5 above we have the following derivation:

$$\begin{aligned} S &\Rightarrow aSBC \\ &\Rightarrow aaBCBC \\ &\Rightarrow aaBBCC \\ &\Rightarrow aabBCC \\ &\Rightarrow aabbCC \\ &\Rightarrow aabbcC \\ &\Rightarrow aabbcc \end{aligned}$$

Therefore we have for instance

$$\begin{aligned} S &\Rightarrow^* aabbcc \\ S &\Rightarrow^* aabBCC \\ aaBCBC &\Rightarrow^* aabBCC \\ aaBCBC &\Rightarrow^* aaBCBC \end{aligned}$$

Remark

Remark

The above definitions introduced relations

$$\begin{aligned} \Rightarrow_G &\subseteq (T \cup N)^+ \times (T \cup N)^* \\ \Rightarrow^*_G &\subseteq (T \cup N)^* \times (T \cup N)^* \end{aligned}$$

$\Rightarrow^*_G$ *is the reflexive and transitive closure of* $\Rightarrow_G$

II.1.3 Grammars and Derivations (10.2)
II.1.3.1. Derivations (10.2.3.)
II.1.3.2. Language Generation (10.2.4.)

Language Generation (10.2.4.)

Definition

Let $G = (T, N, S, P)$ be a grammar. The **language generated by the grammar** G, denoted by $L(G) \subseteq T^*$ is defined as the set of terminal strings generated from the start symbol, i.e.

$$L(G) := \{w \in T^* \mid S \Rightarrow_G^* w\}$$

Remark

If a language is generated by a grammar, then there are infinitely many grammars which generate the same language.

Equivalence of Grammars

Definition

We say that two grammars G_1 and G_2 are **equivalent** iff

$$L(G_1) = L(G_2)$$

Example

We show that

$$L(G^{a^n b^n}) = \{a^n b^n \mid n \geq 1\}$$

One can easily show that

$$S \Rightarrow^* t \Leftrightarrow \exists n \geq 0 (t = a^n S b^n \vee t = a^{n+1} b^{n+1})$$

- "$\Rightarrow$" follows by induction on length of the derivation $S \Rightarrow^* t$.
- "$\Leftarrow$": show first the assertion for $t = a^n S b^n$ by induction on n. Then the assertion for $t = a^{n+1} b^{n+1}$ follows as well.

II.1.1 Administrative Issues

II.1.2 Overview Part II

II.1.3 Grammars and Derivations (10.2)
II.1.3.1. Derivations (10.2.3.)
II.1.3.2. Language Generation (10.2.4.)

II.1.4 The Chomsky Hierarchy (12.1)

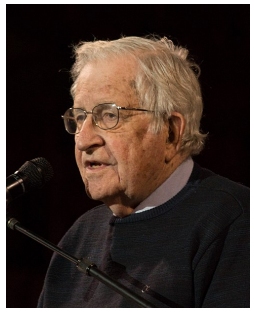
Chomsky Hierarchy

The **Chomsky hierarchy** is the classification of grammars by means of 4 properties of its production rules:

- **Regular grammars (Type-3).**
  - Simple to parse. Used for dividing the input stream of characters into tokens.
- **Context-free grammars (Type-2).**
  - Easy to understand and supported by parse generators.
  - In language design one aims at languages having an underlying context-free grammar.
- **Context-sensitive grammars (Type-1).**
  - It's usually an accident if the basic grammar (excluding features such as variables need to be declared) of a language is context-sensitive. C and C++ have some context-sensitive aspects (dealt with by selecting correct strings after the parsing).
- **Unrestricted grammars (Type-0).**
  - The limit of grammars.

Noam Chomsky

Noam Chomsky
linguist, cognitive scientist, historian,
social critic, and political activist.
Originator of the Chomsky Hierarchy



(Source:
https://en.wikipedia.org/wiki/File:Noam_Chomsky_portrait_2017.jpg)

Examples of Equivalent Grammars

- We give grammars of each type for defining the language

$$L^{a^{2n}} := \{a^i \mid i \text{ is even } \}$$

- Note that it belongs to all 4 types of the Chomsky Hierarchy, especially to the highest one (regular grammars)
- The grammars for the lower types of the Chomsky Hierarchy will be a bit artificial, since they are made more complicated in order not to belong to the higher type ones.

Type-3 Grammars: Regular Grammars

Definition

1. A grammar G is **left-linear**, iff all its productions have the form

$$A \longrightarrow Ba \text{ or } A \longrightarrow a \text{ or } A \longrightarrow \epsilon$$

2. A grammar G is **right-linear**, iff all its productions have the form

$$A \longrightarrow aB \text{ or } A \longrightarrow a \text{ or } A \longrightarrow \epsilon$$

3. A grammar G is of **Type-3** or **regular**, iff it is left-linear or right-linear

In the above we have $A, B \in N$ and $a \in T$.

No Mixing of Left-Linear and Right-Linear

Remark:
In a regular grammar either all productions must be left-linear or all productions must be right-linear, so no mixing of the left-linear and right-linear is allowed.

Remark

- In 2015/16 we allowed Productions of the form $A \longrightarrow B$ (**silent productions**) in regular grammars.
- This turned out to cause technical problems, so we decided to not allow them anymore.
- Our definition follows the traditional definitions of regular grammars.

Left Linear (Regular) Grammar for $L^{a^{2n}}$

grammar	$G^{leftlinear,a^{2n}}$
terminals	a
nonterminals	S, A
start symbol	S
productions	$S \longrightarrow \epsilon$ $S \longrightarrow Aa$ $A \longrightarrow Sa$

Right Linear (Regular) Grammar for $L^{a^{2n}}$

grammar	$G^{rightlinear,a^{2n}}$
terminals	a
nonterminals	S, A
start symbol	S
productions	$S \longrightarrow \epsilon$ $S \longrightarrow aA$ $A \longrightarrow aS$

Complexity

- Regular languages are exactly those languages which can be recognised using a constant space algorithm.
  So the memory used for parsing is linear in the input.
- This algorithm will be an automaton, and will recognise the language in linear time
  - i.e. in time $O(|s|)$, where $|s|$ is the length of the string.

Types-2 Grammars: Context-Free Grammars

Definition

Any grammar G is of **Type-2** or **context-free**, if all its productions have the form

$$A \longrightarrow w$$

where $A \in N$ is a nonterminal, which rewrites to a string $w \in (T \cup N)^*$.

Context-Free Grammar for $L^{a^{2n}}$

grammar	$G^{context-free,a^{2n}}$
terminals	a
nonterminals	S
start symbol	S
productions	$S \longrightarrow \epsilon$ $S \longrightarrow aSa$

Complexity

- Context free languages can be recognised in $O(|s|^4)$ time, where $|s|$ is the length of the input stream.
  If three tapes are allowed they can be recognised in $O(|s|^3)$ time.
- Most practical examples belong to subclasses such as the LL(n)- or LR(n)-languages, which can be recognised in linear time (i.e. $O(|s|)$).

Type 1 Grammars: Context-Sensitive Grammars

Definition

Any grammar G is of **Type-1** or **context-sensitive**, if all its productions have the form

$$uAv \longrightarrow uwv$$

where $A \in N$ is a nonterminal, which rewrites to a non-empty string $w \in (T \cup N)^+$, but only where A is in the context of strings $u, v \in (T \cup N)^*$.
Furthermore a production

$$A \longrightarrow \epsilon$$

is allowed, but only if A does not occur in the right hand side of any production.

Context-Sensitive Grammar for $L^{a^{2n}}$

grammar	$G^{context-sensitive, a^{2n}}$
terminals	a
nonterminals	S, T
start symbol	S
productions	$S \longrightarrow \epsilon$ $S \longrightarrow aa$ $S \longrightarrow aaT$ $aT \longrightarrow aTaa$ $aT \longrightarrow aaa$

Complexity

- The class of context sensitive is PSPACE complete, where PSPACE is the class of languages, which can be recognised with polynomial amount of space.
- The complexity class PSPACE is at least as big as P (polynomial time) and a strict subset of EXPTIME (exponential time).

Type-0 Grammars: Unrestricted Grammars

Let in the following four definitions $G = (T, N, S, P)$ be a grammar.

Definition

Any grammar G is of **Type 0** or **unrestricted**, so any production

$$u \longrightarrow v$$

for $u \in (T \cup N)^+$, $v \in (T \cup N)^*$ are allowed.

Unrestricted Grammar for $L^{a^{2n}}$

grammar	$G^{unrestricted,a^{2n}}$
terminals	a
nonterminals	S
start symbol	S
productions	$S \longrightarrow \epsilon$ $S \longrightarrow aa$ $a \longrightarrow aaa$

Complexity

- The class of unrestricted grammars is equal to the set of **computably enumerable languages** (also called **recursively enumerable** languages).
- This means that we cannot decide whether a string belongs to such a language.
  However there exists an algorithm, which
  - if a string belongs to a language, eventually will accept it,
  - if a string does not belong to the language possibly will run forever.
- Computable and computable enumerable languages will be discussed in part III of this module.

A Hierarchy of Languages

Definition

A language $L \subseteq T^*$ is **regular**, **context-free**, **context-sensitive**, or **unrestricted**, iff there exists a grammar G of the relevant type such that $L(G) = L$.

Remark

For any language L we have L regular $\Rightarrow$ L context-free $\Rightarrow$ L context-sensitive $\Rightarrow$ L unrestricted.

Common Mistake in Exams

Warning In the exam we often see students writing "this language is not context free because it is regular". This is **wrong.** Regular languages are a subset of the context free languages, any regular language is as well context free.

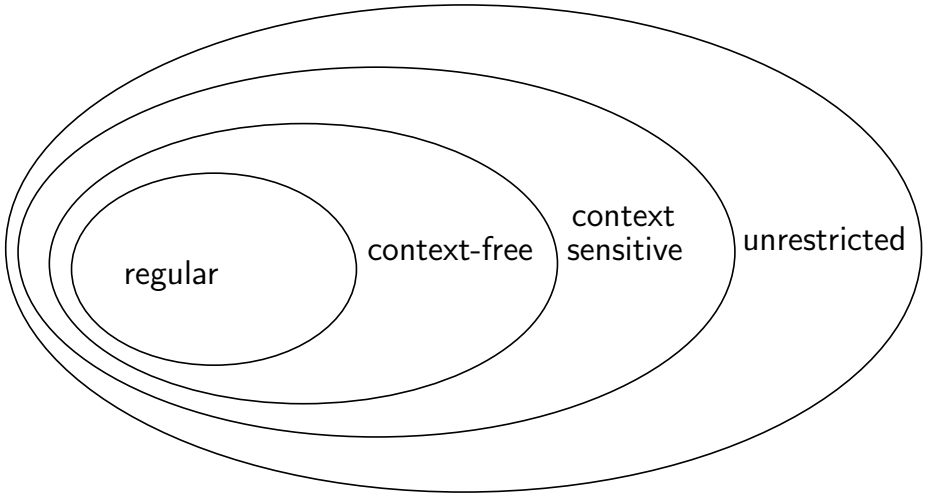
Hierarchy of Languages

We have that

- every regular grammar is context-free.
- every context-sensitive grammar is an unrestricted grammar.

However not every context-free grammar is context sensitive, since context-sensitive languages allow only productions $A \longrightarrow \epsilon$ if A does not occur at the right hand side of a production. (Otherwise all unrestricted languages would be context-sensitive).
However one can construct from a context-free grammar a context-free grammar of the same language, which has only productions $A \longrightarrow \epsilon$, if A does not occur on the right hand side of a production. This grammar is therefore context-sensitive as well.

Hierarchy of Languages



Example 1 (Grammars of the Types of the Chomsky Hierarchy)

grammar	G
terminals	a, b
nonterminals	S
start symbol	S
productions	$S \longrightarrow \mathrm{a}S\mathrm{a}, S \longrightarrow \mathrm{b}S\mathrm{b}, S \longrightarrow \epsilon$

$L(G) = ?$
G is of which type?

Example 2

grammar	G
terminals	a
nonterminals	S
start symbol	S
productions	$S \longrightarrow \mathrm{a}, S \longrightarrow \mathrm{a}S$

$L(G) = ?$
G is of which type?

Example 3

grammar	G
terminals	a, b
nonterminals	S
start symbol	S
productions	$S \longrightarrow \text{ab}, S \longrightarrow \text{a}S\text{b}$

$L(G) = ?$
G is of which type?

Example 4 (Grammars)

Consider the grammar

grammar	$G^{a^n b^n c^n}$
terminals	a, b, c
nonterminals	S, B, C
start symbol	S
productions	$S \longrightarrow aSBC, S \longrightarrow aBC,$ $CB \longrightarrow BC,$ $aB \longrightarrow ab, bB \longrightarrow bb, bC \longrightarrow bc, cC \longrightarrow cc.$

Example 4 is not a context sensitive grammar
(although often referred to as context sensitive). Why?

Example 5 (Grammars)

Here is a variant which is context sensitive:

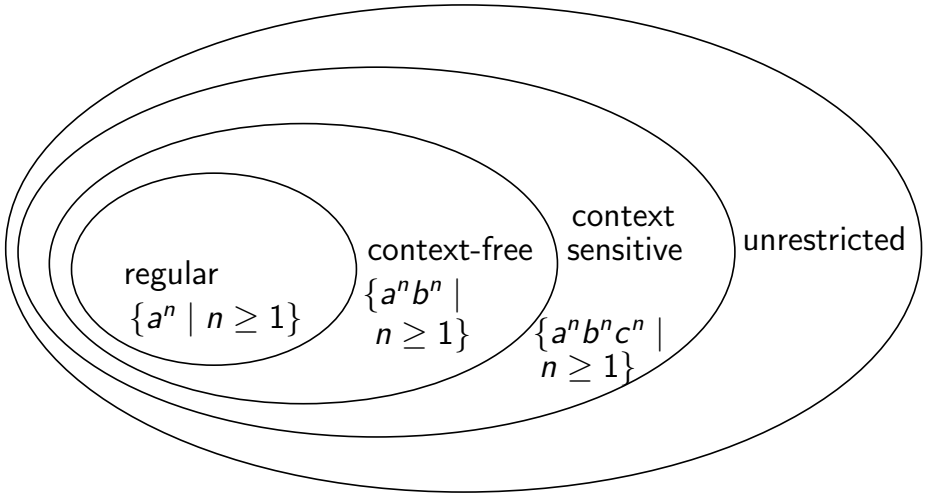
grammar	$G^{a^n b^n c^n}$
terminals	a, b, c
nonterminals	S, B, C, H
start symbol	S
productions	$S \longrightarrow aSBC, S \longrightarrow aBC,$ $CB \longrightarrow HB, HB \longrightarrow HC, HC \longrightarrow BC,$ $aB \longrightarrow ab, bB \longrightarrow bb, bC \longrightarrow bc, cC \longrightarrow cc.$

Example Derivation

A derivation of $aabbcc$ in Example 5 is as follows:

$$\begin{aligned} S &\Rightarrow aSBC \\ &\Rightarrow aaBCBC \\ &\Rightarrow aaBHBC \\ &\Rightarrow aaBHCC \\ &\Rightarrow aaBBCC \\ &\Rightarrow aabBCC \\ &\Rightarrow aabbCC \\ &\Rightarrow aabbcC \\ &\Rightarrow aabbcc \end{aligned}$$

Examples


regular
$\{a^n \mid n \geq 1\}$
context-free
$\{a^n b^n \mid n \geq 1\}$
context
sensitive
$\{a^n b^n c^n \mid n \geq 1\}$
unrestricted