Final lecture: Chomsky hierarcy, applications, and recap

Twan van Laarhoven

Institute for Computing and Information Sciences – Intelligent Systems
Radboud University Nijmegen

Version: fall 2014

Outline

Questions about context-free languages

Decidable (general algorithm exists)

- Given any CFG G and $w \in \Sigma^*$, is $w \in \mathcal{L}(G)$? (build PDA).
- Given PDA M, is $\mathcal{L}(M) = \emptyset$?
- Given PDA M, is $\mathcal{L}(M)$ finite?
- Given any CFG G, is $\mathcal{L}(G)$ regular?

Undecidable (no general algorithm exists)

- Given any CFG G, is $\mathcal{L}(G) = \Sigma^*$?
- Given any CFGs G_1 and G_2, is $\mathcal{L}(G_1) = \mathcal{L}(G_2)$?
- Given any CFG G, is $\mathcal{L}(G)$ deterministic?
- Given any CFG G, is it ambiguous?

Pumping lemma

Lemma: If L is context-free, then there exists a $k \geq 1$, such that every $s \in L$ with $|s| \geq k$ can be written as $s = uvwxy$. With

- $|vwx| \leq k$,
- $|vx| \geq 1$, and
- $uv^n wx^n y \in L$ for all $n \geq 0$.

Intuition: derivation trees.

Example: $L_1 = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}$

Context-sensitive languages

A context-free grammar has rules like:

$$X \quad \rightarrow \quad w$$

With $X \in V$, $w \in (\Sigma \cup V)^*$.

A context-sensitive grammar has rules like:

$$\alpha X \beta \quad \rightarrow \quad \alpha w \beta$$

With $X \in V$, $\alpha, \beta, w \in (\Sigma \cup V)^*$, $w \neq \lambda$.

Context-sensitive grammars generate context-sensitive languages.

Context-sensitive languages

Example: $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n > 0\}$

$$
\begin{array}{lcl}
S & \to & \mathtt{a}BC \mid \mathtt{a}SBC \\
CB & \to & XB \\
XB & \to & XC \\
XC & \to & BC \\
\mathtt{a}B & \to & \mathtt{ab} \\
\mathtt{b}C & \to & \mathtt{bc}
\end{array}
$$

Enumerable and computable languages

An unrestricted grammar has rules like:

$u \rightarrow v$

With $u, v \in (\Sigma \cup V)^*$.

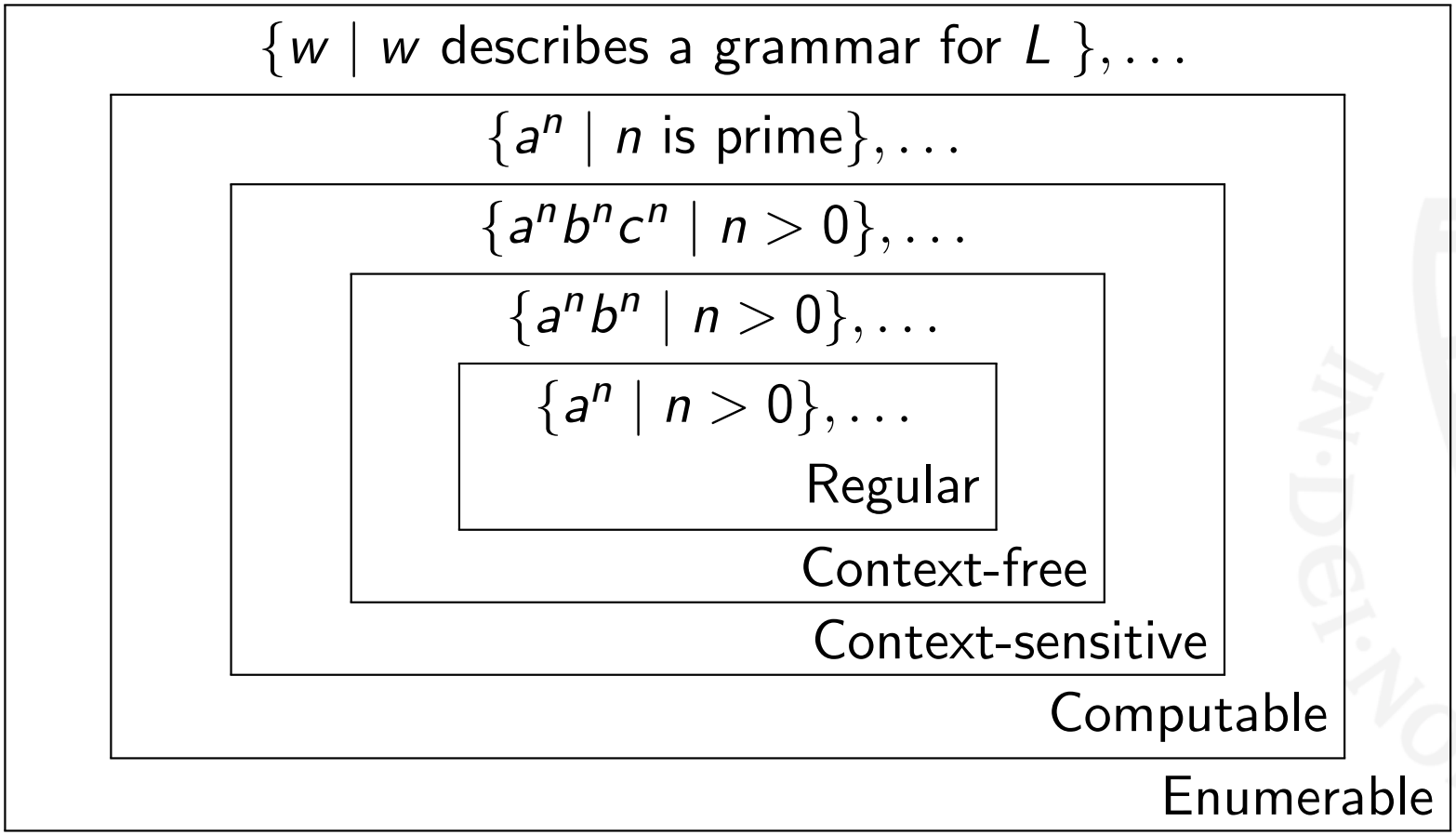
Recognized by Turing machines (bidirectional tape) or queue automaton.

Unrestricted grammars generate enumerable languages.

A languages is computable if both L and $\overline{L} = \Sigma^* - L$ are enumerable.

Example: $\{\mathtt{a}^n \mid n \text{ is not prime}\}$.

Chomsky hierarchy



Trade-offs

Bigger classes of languages:

- More languages can be described.
- But, you can say less about them.

	$w \in L$?	time	memory	$L_1 = L_2$?
Regular	yes	$\lvert w \rvert$	const.	yes
Deterministic context-free	yes	$\lvert w \rvert$	$\lvert w \rvert$	no
Context-free	yes	$\lvert w \rvert^3$	$\lvert w \rvert^2$	no
Context-sensitive	yes	$2^{\lvert w \rvert}$	$\lvert w \rvert^k$	no
Computable	yes	∞	∞	no
Enumerable	if $w \in L$	∞	∞	no

Programming languages

Most programming languages are (deterministic) context-free.

There are tools to automatically build:

- a lexical analyzer ('lexer') from regular expressions.
- a parser from a CFG.

Natural language

$$
\begin{array}{rcl}
S = \langle \text{sentence} \rangle & \to & \langle \text{noun-phrase} \rangle \langle \text{verb-phrase} \rangle . \\
\langle \text{sentence} \rangle & \to & \langle \text{noun-phrase} \rangle \langle \text{verb-phrase} \rangle \langle \text{object-phrase} \rangle . \\
\langle \text{noun-phrase} \rangle & \to & \langle \text{name} \rangle \langle \text{article} \rangle \langle \text{noun} \rangle \\
\langle \text{name} \rangle & \to & \texttt{John} \mid \texttt{Jill} \\
\langle \text{noun} \rangle & \to & \texttt{bicycle} \mid \texttt{mango} \\
\langle \text{article} \rangle & \to & \texttt{a} \mid \texttt{the} \\
\langle \text{verb-phrase} \rangle & \to & \langle \text{verb} \rangle \mid \langle \text{adverb} \rangle \langle \text{verb} \rangle \\
\langle \text{verb} \rangle & \to & \texttt{eats} \mid \texttt{rides} \\
\langle \text{adverb} \rangle & \to & \texttt{slowly} \mid \texttt{frequently} \\
\langle \text{adjective-list} \rangle & \to & \langle \text{adjective} \rangle \langle \text{adjective-list} \rangle \mid \lambda \\
\langle \text{adjective} \rangle & \to & \texttt{big} \mid \texttt{juicy} \mid \texttt{yellow} \\
\langle \text{object-phrase} \rangle & \to & \langle \text{adjective-list} \rangle \langle \text{name} \rangle \\
\langle \text{object-phrase} \rangle & \to & \langle \text{article} \rangle \langle \text{adjective-list} \rangle \langle \text{noun} \rangle
\end{array}
$$

Jill frequently eats a juicy yellow mango. belongs to this language

Lindenmayer systems

Model of organism growth.

Example:

$$\begin{array}{lcl} A & \to & AB \\ B & \to & A \end{array}$$

Start symbol A.

Expand once per iteration

0	A
1	AB
2	ABA
3	$ABAAB$
4	$ABAABABA$
	$\ldots$

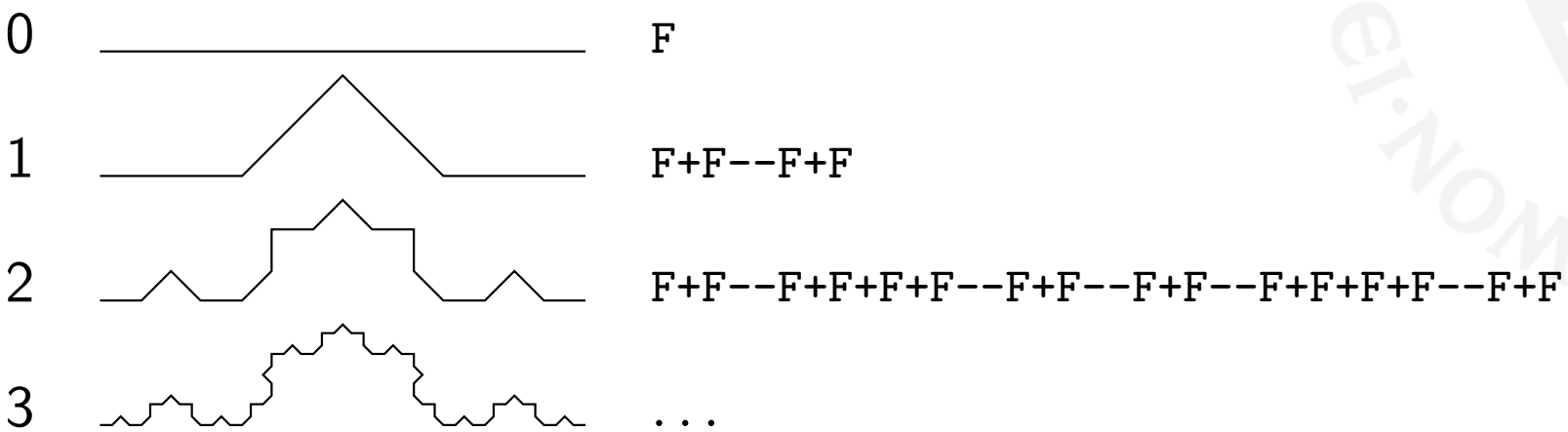
Lindenmayer systems

Drawing a Lindenmayer system:

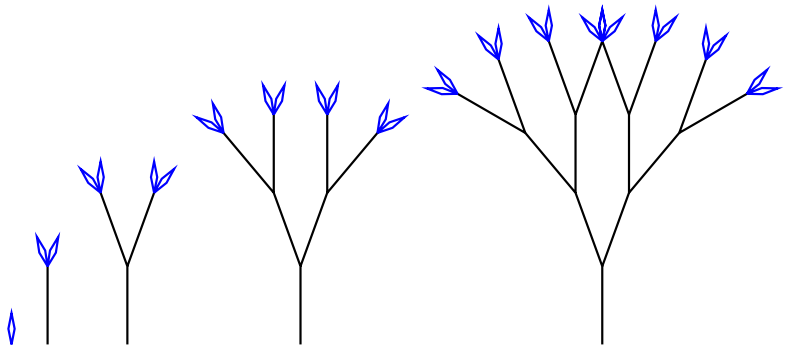
- F: move forward
- +: rotate clockwise
- −: rotate counter clockwise
- [···]: branch

Example:

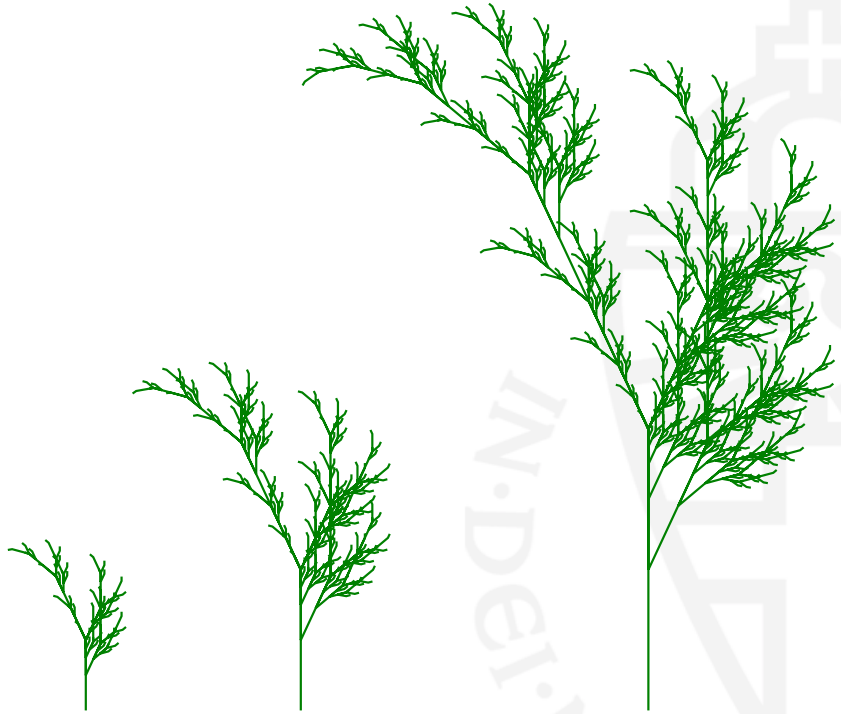
$$F \rightarrow F + F - - F + F$$



Lindenmayer systems



$$S \rightarrow F[+S][-S]$$



$$\begin{array}{lcl} S & \rightarrow & F-[[S]+S]+F[+FS]-S \\ F & \rightarrow & FF \end{array}$$

Remarks on sets

Beware that $\emptyset \neq \{\emptyset\}$ $\emptyset \neq \lambda$ $\emptyset \neq \{\lambda\}$.
and $\emptyset \cdot L = L \cdot \emptyset = \emptyset$
and $\{\lambda\} \cdot L = L \cdot \{\lambda\} = L$.

Symbols:

- $w \in L$: "is in".
- $L_1 \subseteq L_2$: is a subset of, i.e. everything in L_1 is also in L_2.
- $L_1 \cup L_2$: union, the things in either of the two sets.
- $L_1 \cap L_2$: intersection, only the things in both sets.
- $\overline{L}$: complement, the words not in L, $\overline{L} = \Sigma^* - L$.

Terminology: Language described using set-notation, examples:

$$\{w \in \{a,b\}^* \mid |w|_a \text{ is even}\}, \quad \{a^n b^m \mid n < m\}, \quad \emptyset$$

Remarks on words and languages

Languages

- *contain* words.
- can be infinite, but words are finite.

The language L^*

- always contains λ.
- is *not* the same as $\{w^n \mid w \in L, n \geq 0\}$.
- is $L^0 \cup L^1 \cup L^2 \cup \cdots$.

Remarks on proofs

To prove that $L_1 = L_2$, show that

- $L_1 \subseteq L_2$ (for all w, $w \in L_1$ implies $w \in L_2$), and
- $L_1 \supseteq L_2$ (for all w, $w \in L_2$ implies $w \in L_1$).

Proof by contradiction

- If regular languages have property X, and L_1 does not have property X, then L_1 is not regular.
- If from L_1 being context-free you can deduce that $\{a^n b^n c^n \mid n \geq 0\}$ is context-free, then L_1 is not context-free.

Proof by induction: to prove $P(w)$ for all w,

- Show that $P(\lambda)$
- And that $P(w)$ implies $P(aw)$.

Exam topic: regular expressions

You should know:

- the definition of regular expression.
- how to compute the language denoted by a regular expr.
- how to specify a given language with a regular expression (!).
- how to compute a regular expression from an NFA_λ using the elimination-of-states method (!).

(!): Noted as general problem during midterm test.

Exam topic: automata

You should know:

- the definition of DFA, NFA, NFA_λ.
- how to construct a DFA, NFA or NFA_λ for a given language.
- how to draw an automaton. (!).
- how to perform λ-elimination and the subset construction.
- the constructions on DFAs for complement and intersection (product construction) of languages (!).
- the constructions on NFA_λ's for union, concatenation and star of languages.

Exam Topic: regular languages

You should know:

- how to show that two languages are equal (!).
- the closure properties of regular languages, and how to use them to prove that a language is regular (or is not regular) (!).
- Kleene's theorem.
- how to apply the pumping lemma to show that a language is not regular (!).

Exam topic: grammars

You should know:

- the definition of context-free grammar (CFG).
- how to generate strings in a CFG (with leftmost derivations).
- how to find the language generated by a (simple) CFG.
- how to construct a CFG that generates a given (context-free) language.
- the definition of regular grammar.
- how to construct a regular grammar from a NFA.
- how to construct an NFA_λ from a regular grammar.

Exam topic: pushdown automata and CFLs

You should know:

- the definition of pushdown automata (PDAs).
- how to give a PDA for a given (simple context-free) language.
- how find the language accepted by a given PDA.
- how to construct a PDA from a CFG.
- the closure properties of context-free languages.

Exam tips

- You may use results and examples we treated during the lectures and exercises. For example, you may use that the language $\{a^n b^n \mid n \geq 0\}$ is not regular without re-proving it.
- Always give an explanation. For example, when asked to give a CFG for a language, explain why your CFG is correct.
- Check your results. For example,
  - check that a DFA that you give indeed is a DFA.
  - given NFA-λ that accepts w, after λ-elimination and subset constructon, check that resulting DFA accepts w.
- Connect your knowledge, think further. An exam question may not directly tell you what you need to do. For example,
  - Q: Is L regular? (Which techniques can you apply?)
  - Q: Give a DFA for L (Is L of the form $\overline{L_1}$ or $L_1 \cap L_2$?)

Finally

- Register for the exam!
- Vragenuurtje: Fri 23 jan. 2015, 13:30 – 15.30 in HG.00.633