The Chomsky Hierarchy

Formal Grammars, Languages, and the Chomsky-Schützenberger Hierarchy

Overview

- 01 Personalities
- 02 Grammars and languages
- 03 The Chomsky hierarchy
- 04 Conclusion

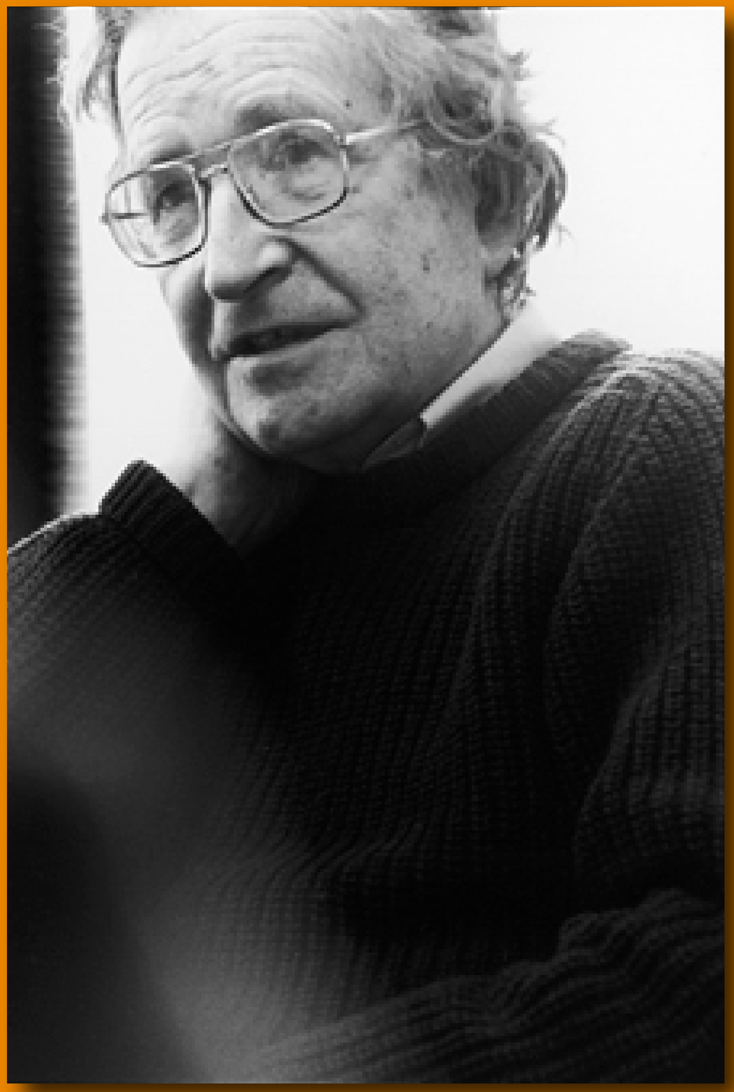
Personalities

Noam Chomsky
Marcel Schützenberger
Others...

01

Noam Chomsky

- Born December 7, 1928
- Currently Professor Emeritus of linguistics at MIT
- Created the theory of generative grammar
- Sparked the cognitive revolution in psychology



Noam Chomsky

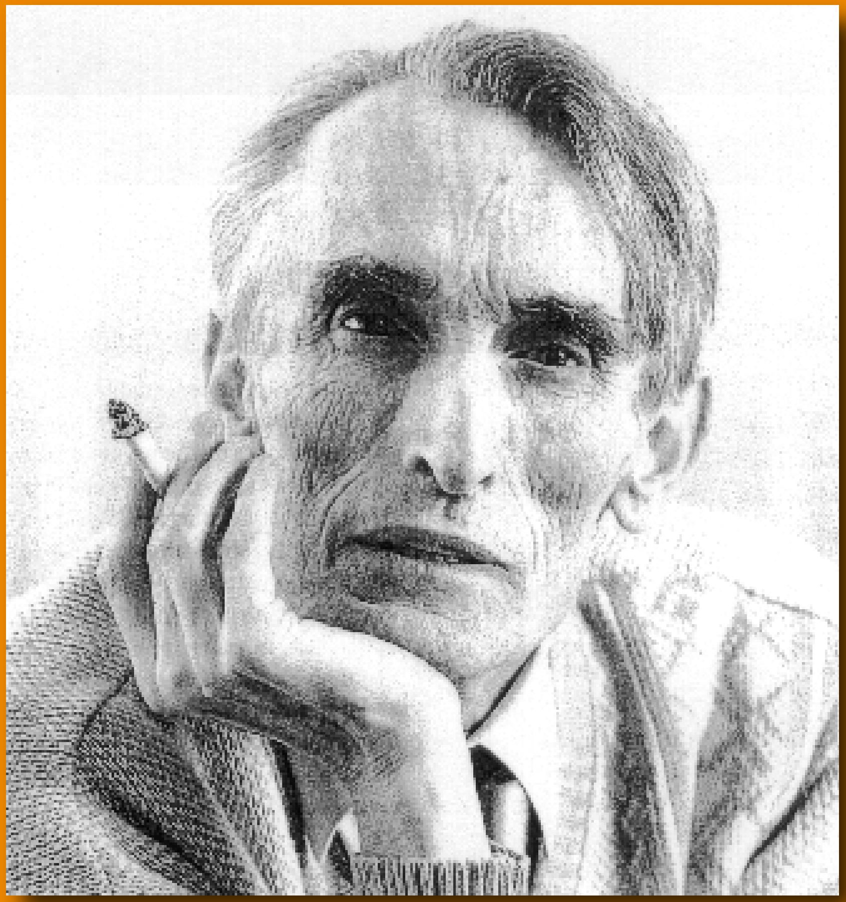
- From 1945, studied philosophy and linguistics at the University of Pennsylvania
- PhD in linguistics from University of Pennsylvania in 1955
- 1956, appointed full Professor at MIT, Department of Linguistics and Philosophy
- 1966, Ferrari P. Ward Chair; 1976, Institute Professor; currently Professor Emeritus

Contributions

- Linguistics
  - Transformational grammars
  - Generative grammar
  - Language aquisition
- Computer Science
  - Chomsky hierarchy
  - Chomsky Normal Form
  - Context Free Grammars
- Psychology
  - Cognitive Revolution (1959)
  - Universal grammar

Marcel-Paul Schützenberger

- Born 1920, died 1996
- Mathematician, Doctor of Medicine
- Professor of the Faculty of Sciences, University of Paris
- Member of the Academy of Sciences



Marcel-Paul Schützenberger

- First trained as a physician, doctorate in medicine in 1948
- PhD in mathematics in 1953
- Professor at the University of Poitiers, 1957-1963
- Director of research at the CNRS, 1963-1964
- Professor in the Faculty of Sciences at the University of Paris, 1964-1996

Contributions

- Formal languages with Noam Chomsky
  - Chomsky-Schützenberger hierarchy
  - Chomsky-Schützenberger theorem
- Automata with Samuel Ellenberger
- Biology and Darwinism
  - Mathematical critique of neo-darwinism (1966)

Grammars and languages

Definitions

Languages and grammars

Syntax and sematics

02

Definitions

Definitions

Noam Chomsky, *On Certain Formal Properties of Grammars*, Information and Control, Vol 2, 1959

Definitions

- **Language**: “A language is a collection of sentences of finite length all constructed from a finite alphabet of symbols.”

Noam Chomsky, *On Certain Formal Properties of Grammars*, Information and Control, Vol 2, 1959

Definitions

- **Language**: “A language is a collection of sentences of finite length all constructed from a finite alphabet of symbols.”
- **Grammar**: “A grammar can be regarded as a device that enumerates the sentences of a language.”

Noam Chomsky, *On Certain Formal Properties of Grammars*, Information and Control, Vol 2, 1959

Definitions

- **Language**: “A language is a collection of sentences of finite length all constructed from a finite alphabet of symbols.”
- **Grammar**: “A grammar can be regarded as a device that enumerates the sentences of a language.”
- A grammar of L can be regarded as a function whose range is exactly L

Noam Chomsky, *On Certain Formal Properties of Grammars*, Information and Control, Vol 2, 1959

Types of grammars

- **Prescriptive** prescribes authoritative norms for a language
- **Descriptive** attempts to describe actual usage rather than enforce arbitrary rules
- **Formal** a precisely defined grammar, such as context-free
- **Generative** a formal grammar that can “generate” natural language expressions

Formal grammars

- Two broad categories of formal languages: **generative** and **analytic**
- A generative grammar formalizes an algorithm that generates valid strings in a language
- An analytic grammar is a set of rules to reduce an input string to a boolean result that indicates the validity of the string in the given language.
- A generative grammar describes how to *write* a language, and an analytic grammar describes how to *read* it (a parser).

- Chomsky posits that each sentence in a language has two levels of representation: **deep structure** and **surface structure**
- Deep structure is a direct representation of the semantics underlying the sentence
- Surface structure is the syntactical representation
- Deep structures are mapped onto surface structures via *transformations*

Transformational grammars

usually synonymous with the more specific transformational-generative grammar (TGG)

Formal grammar

- A formal grammar is a quad-tuple $G = (N, \Sigma, P, S)$ where
  - N is a finite set of non-terminals
  - Σ is a finite set of terminals and is disjoint from N
  - P is a finite set of production rules of the form

    $w \in (N \cup \Sigma)* \longrightarrow w \in (N \cup \Sigma)*$
  - $S \in N$ is the start symbol

The Chomsky hierarchy

Overview
Levels defined
Application and benefit

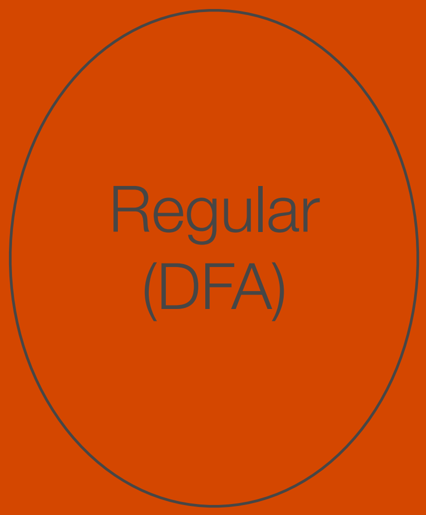
03

The hierarchy

- A containment hierarchy (strictly nested sets) of classes of formal grammars

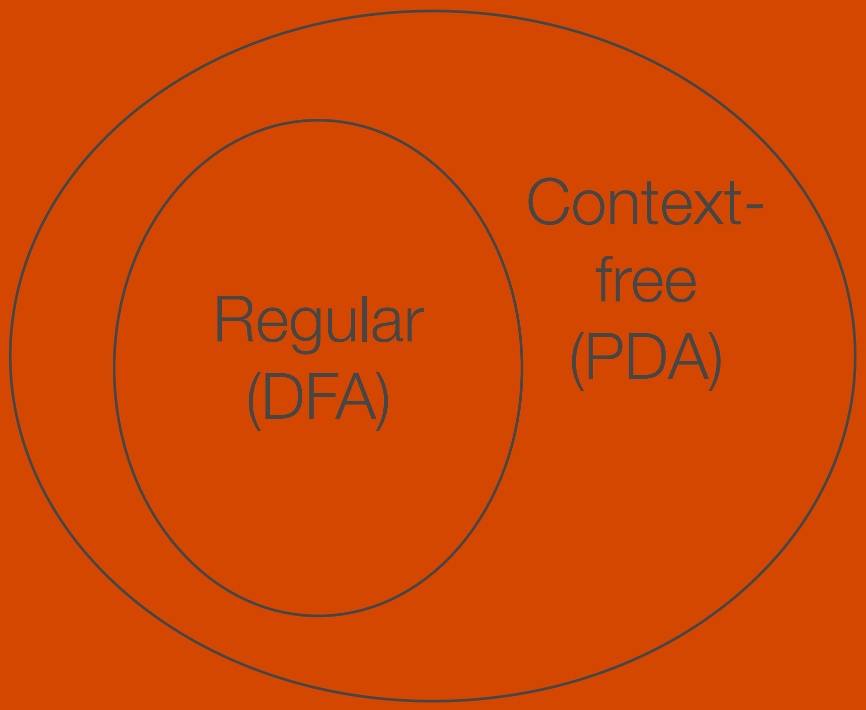
The hierarchy

- A containment hierarchy (strictly nested sets) of classes of formal grammars



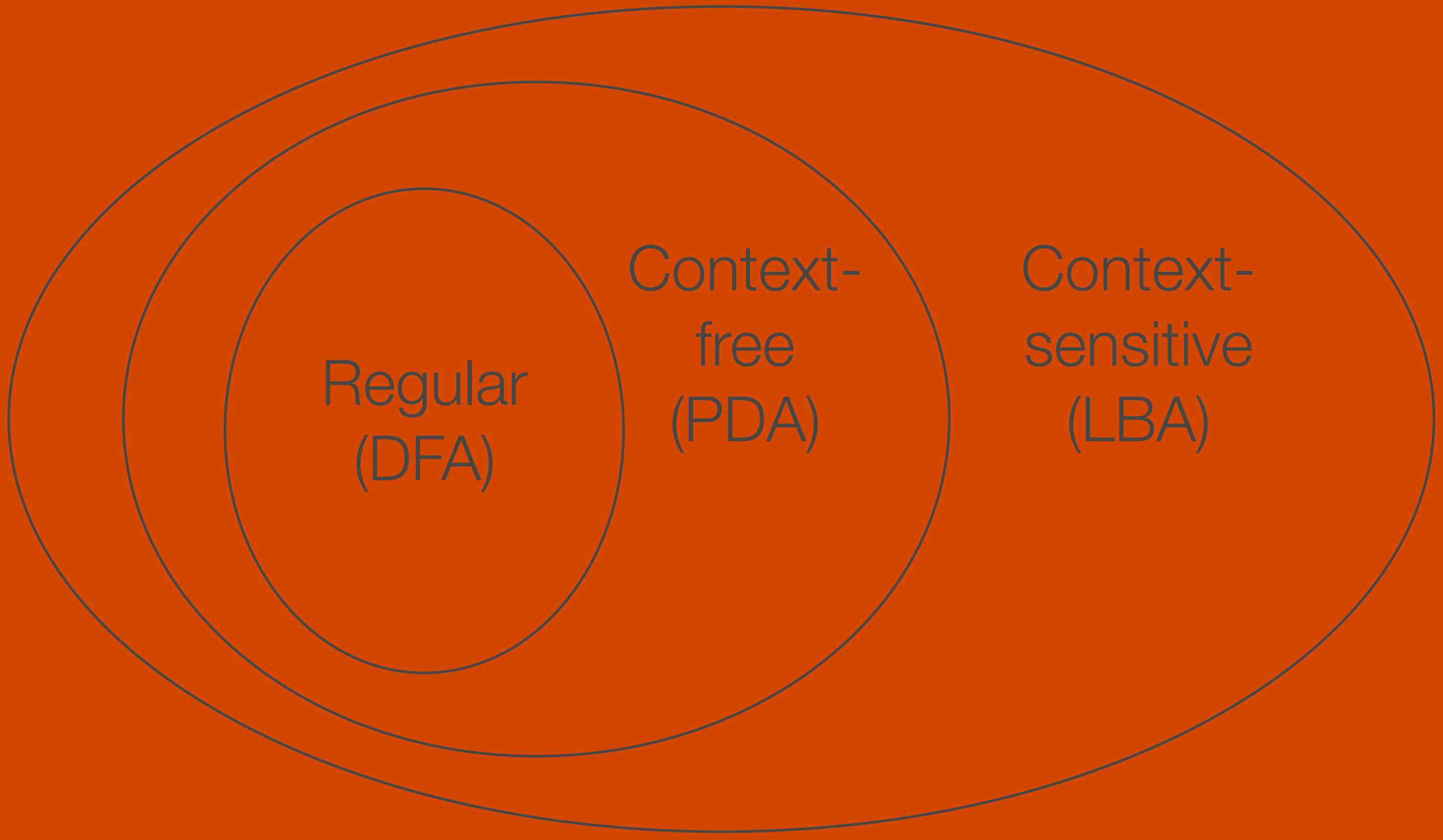
The hierarchy

- A containment hierarchy (strictly nested sets) of classes of formal grammars



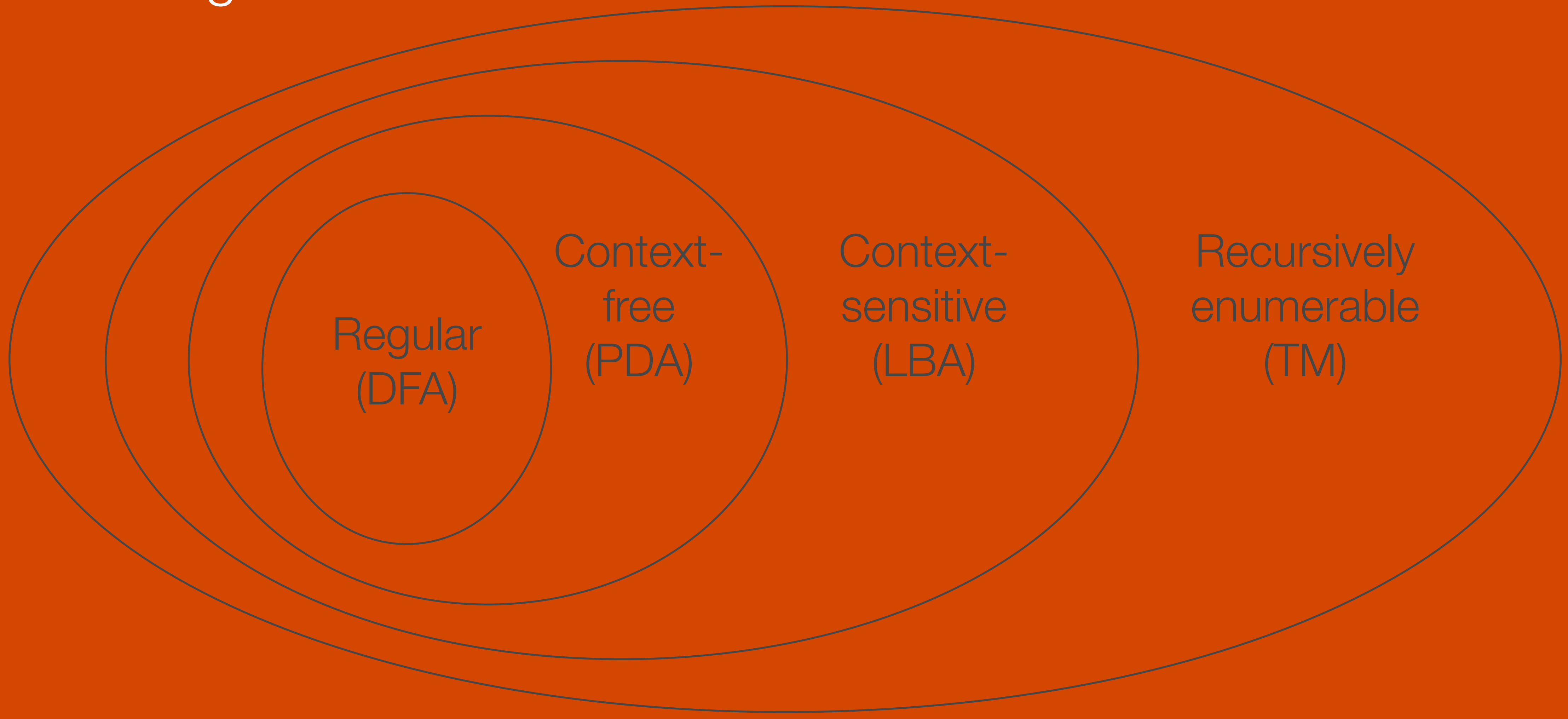
The hierarchy

- A containment hierarchy (strictly nested sets) of classes of formal grammars



The hierarchy

- A containment hierarchy (strictly nested sets) of classes of formal grammars



The hierarchy

The hierarchy

Class	Grammars	Languages	Automaton

The hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursively enumerable (Turing-recognizable)	Turing machine

The hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursively enumerable (Turing-recognizable)	Turing machine
Type-1	Context-sensitive	Context-sensitive	Linear-bounded

The hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursively enumerable (Turing-recognizable)	Turing machine
Type-1	Context-sensitive	Context-sensitive	Linear-bounded
Type-2	Context-free	Context-free	Pushdown

The hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursively enumerable (Turing-recognizable)	Turing machine
Type-1	Context-sensitive	Context-sensitive	Linear-bounded
Type-2	Context-free	Context-free	Pushdown
Type-3	Regular	Regular	Finite

The hierarchy

The hierarchy

Class	Grammars	Languages	Automaton

The hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursively enumerable (Turing-recognizable)	Turing machine

The hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursively enumerable (Turing-recognizable)	Turing machine
	none	Recursive (Turing-decidable)	Decider

The hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursively enumerable (Turing-recognizable)	Turing machine
	none	Recursive (Turing-decidable)	Decider
Type-1	Context-sensitive	Context-sensitive	Linear-bounded

The hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursively enumerable (Turing-recognizable)	Turing machine
	none	Recursive (Turing-decidable)	Decider
Type-1	Context-sensitive	Context-sensitive	Linear-bounded
Type-2	Context-free	Context-free	Pushdown

The hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursively enumerable (Turing-recognizable)	Turing machine
	none	Recursive (Turing-decidable)	Decider
Type-1	Context-sensitive	Context-sensitive	Linear-bounded
Type-2	Context-free	Context-free	Pushdown
Type-3	Regular	Regular	Finite

Type 0

Unrestricted

- Languages defined by Type-0 grammars are accepted by Turing machines
- Rules are of the form: $\alpha \rightarrow \beta$, where α and β are arbitrary strings over a vocabulary V and $\alpha \neq \varepsilon$

Type 1

Context-sensitive

- Languages defined by Type-0 grammars are accepted by linear-bounded automata
- Syntax of some natural languages (Germanic)
- Rules are of the form:

  $\alpha A\beta \rightarrow \alpha B\beta$

  $S \rightarrow \varepsilon$

  where

  $A, S \in N$

  $\alpha, \beta, B \in (N \cup \Sigma)*$

  $B \neq \varepsilon$

Type 2

Context-free

- Languages defined by Type-2 grammars are accepted by push-down automata
- Natural language is almost entirely definable by type-2 tree structures
- Rules are of the form:

  $$A \rightarrow \alpha$$

  where

  $$A \in N$$

  $$\alpha \in (N \cup \Sigma)*$$

- Languages defined by Type-3 grammars are accepted by finite state automata
- Most syntax of some informal spoken dialog
- Rules are of the form:

$$A \rightarrow \varepsilon$$
$$A \rightarrow \alpha$$
$$A \rightarrow \alpha B$$

where

$$A, B \in N \text{ and } \alpha \in \Sigma$$

Type 3

Regular

Programming languages

- The syntax of most programming languages is context-free (or very close to it)
  - EBNF / ALGOL 60
- Due to memory constraints, long-range relations are limited
- Common strategy: a relaxed CF parser that accepts a superset of the language, invalid constructs are filtered
- Alternate grammars proposed: indexed, recording, affix, attribute, van Wijngaarden (VW)

Conclusion

Conclusion
References
Questions

04

Why?

- Imposes a logical structure across the language classes
- Provides a basis for understanding the relationships between the grammars

References

Noam Chomsky, *On Certain Formal Properties of Grammars*, Information and Control, Vol 2 (1959), 137-167

Noam Chomsky, *Three models for the description of language*, IRE Transactions on Information Theory, Vol 2 (1956), 113-124

Noam Chomsky and Marcel Schützenberger, *The algebraic theory of context free languages*, Computer Programming and Formal Languages, North Holland (1963), 118-161

Further information

Wikipedia entry on Chomsky hierarchy and Formal grammars
http://en.wikipedia.org/wiki/Chomsky–Schützenberger_hierarchy
http://en.wikipedia.org/wiki/Formal_grammar

Programming Language Concepts (section on Recursive productions and grammars)
http://www.cs.rit.edu/~afb/20013/plc/slides/

Introduction to Computational Phonology
http://www.spectrum.uni-bielefeld.de/Classes/Winter97/IntroCompPhon/compphon/

Questions

Comments