Abstract: Computational Complexity theory deals with the classification of problems into classes of hardness called complexity classes. In Abstract Complexity (in contrast to Concrete Complexity) we define complexity classes according to general structural properties, such as the model of computation (Turing Machine, RAM, Finite Automaton, PDA, LBA, PRAM, monotone circuits), the mode of computation (deterministic, nondeterministic, probabilistic, alternating, uniform parallel), the kind of the automaton (decider, acceptor, generator, transducer, counting), the resources (time, space, # of processors, circuit size and depth) and also randomness, oracles, interactivity, promise, advice, operators. Inclusion and separation relations between complexity classes constitute central research goals and form some of the most important open questions in theory. This research has led to definitions of scores of complexity classes, as well as sequences of classes known as “complexity hierarchies”. We will review some of the most interesting ones, including the Polynomial-Time Hierarchy, a Counting Hierarchy and an Approximability Hierarchy.

30's, 40's: Unsolvability
Gödel, Kleene

$$\begin{tikzcd}
 & \mathsf{ANALYTICAL} & \\
 & \mathsf{ARITHMETICAL} \arrow[u] & \\
\mathsf{coRE} \arrow[ur] & & \mathsf{RE} \arrow[ul] \\
 & \mathsf{REC} \arrow[ul] \arrow[ur] & \\
 & \mathsf{PrimREC} \arrow[u] & \\
 & \mathsf{Elementary} \arrow[u] &
\end{tikzcd}$$

Kalmar Elementary:

Loop-Computable with number of nested for-loops ≤ 2

PrimREC: Primitive Recursive, Loop-Computable

REC: Recursive, Decidable, Computable

RE: Recursively Enumerable, Listable, Acceptable

ARITHMETICAL: Definable in Arithmetic: $\mathbb{N} = \langle N; <; S; +; *; 0 \rangle$.

Definable by first-order quantified formula over a recursive predicate.

E.g.: $\exists x_1 \forall x_2 \exists x_3 \ldots R(x_1, \ldots, x_k) \in \Sigma_k^0$

ANALYTICAL: Definable by a second-order quantified formula.

E.g., $\exists$ set A, $\forall$ function f, $\ldots$

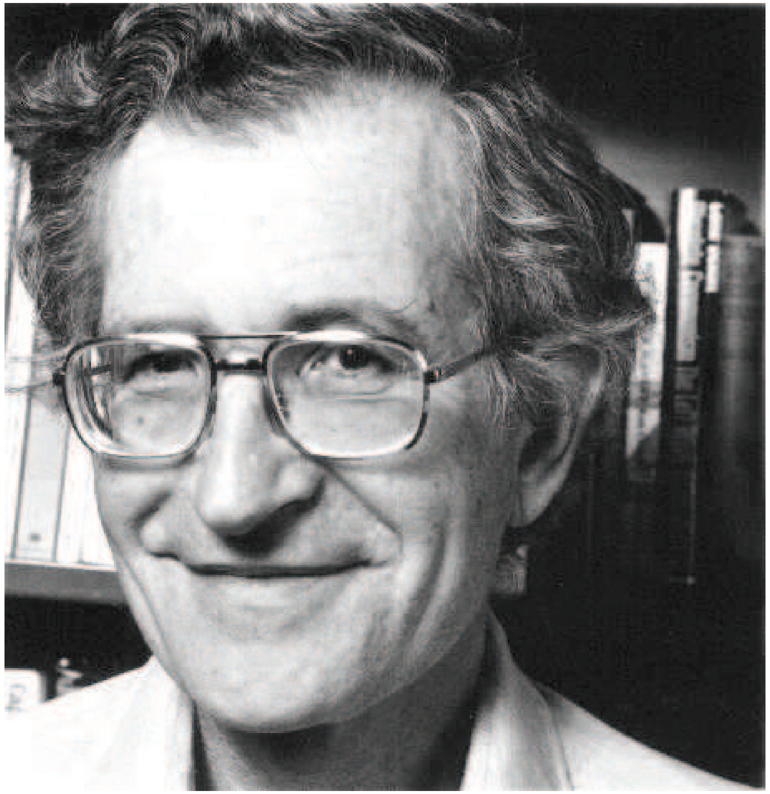
50's: Formal Languages and Automata

Chomsky

Deterministic vs. Nondeterministic Model

Relation of C with coC

$$\begin{array}{ccl}
\mathsf{RE} & & \neq \mathsf{coRE} \\
\uparrow & & \\
\mathsf{CS} & & = \mathsf{coCS} \\
\uparrow & & \\
\mathsf{CF} & & \neq \mathsf{coCF} \\
\uparrow & & \\
\mathsf{REG} & & = \mathsf{coREG} \\
\uparrow & & \\
\mathsf{FIN} & & \neq \mathsf{coFIN}
\end{array}$$



FIN: finite

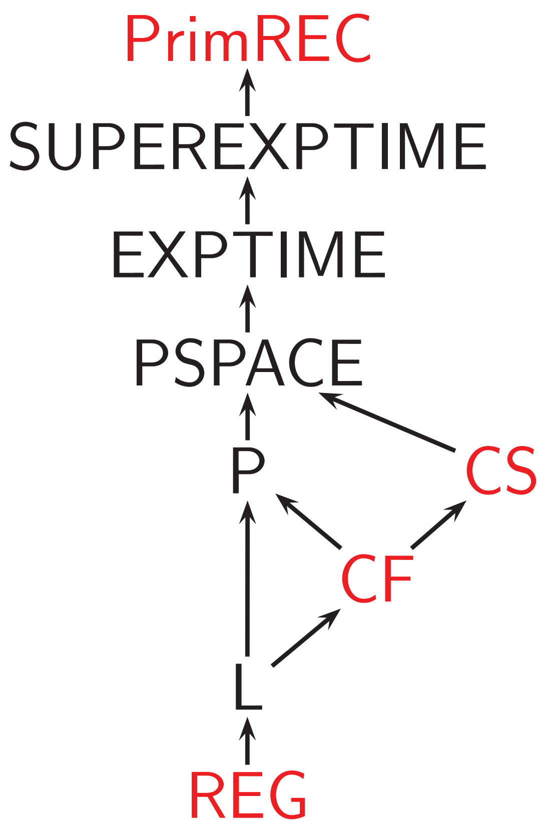
REG: decidable (acceptable) by a (Deterministic or Nondeterministic) Finite Automaton, equivalently definable by a Regular Expression, equivalently generatable by a Right-Linear Grammar

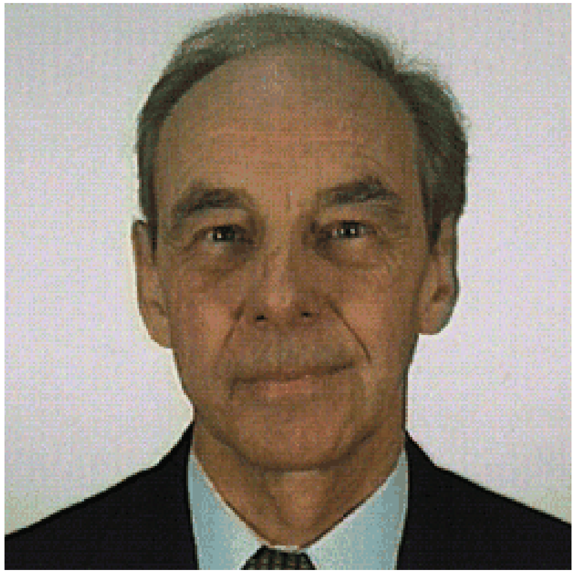
CF: decidable (acceptable) by a (Nondeterministic) Push-Down Automaton, equivalently generatable by a Context-Free Grammar

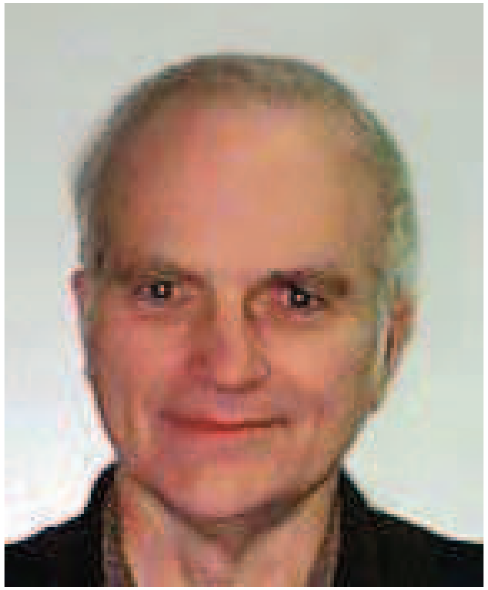
CS: decidable (acceptable) by a (Nondeterministic) Linearly-Bounded Automaton, equivalently generatable by a Context-Sensitive Grammar

RE: acceptable by a (Deterministic or Nondeterministic) Turing Machine, equivalently generatable by a General Grammar

60's: Computational Complexity (Space, Time) Hartmanis







early 70's: Nondeterminism and Complexity

NP-completeness

Cook, Karp, Savitch

$$\begin{array}{c} \mathsf{PSPACE} = \mathsf{NPSPACE} \\ \uparrow \\ \mathsf{NP} \\ \uparrow \\ \mathsf{P} \\ \uparrow \\ \mathsf{NL} \\ \uparrow \\ \mathsf{L} \end{array}$$