# Classical Complexity Classes

Édouard Bonnet Codex 5.6 and 6

**Abstract**

We define classical complexity classes for languages of finite binary strings, using explicit time bounds, work-space bounds, and certificate encodings. The definition of P and its proved finite-stack and single-tape characterizations are included locally. We prove L $\subseteq$ NL $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE = NPSPACE $\subseteq$ EXPTIME, as well as the stated encoding properties and closure of P under complement.

## 1 Languages, machines, and complexity classes

All languages are subsets of the finite binary strings. The following definition text is reproduced from the concept layer. The annotations link it to the precise machine data, encodings, quantifiers, and bounds in Lean.

**Definition 1** (The complexity class P)**.** A language of finite binary strings belongs to P if a deterministic Turing machine decides membership in that language in polynomial time. Precisely, there are a single machine and a polynomial $p \in \mathbb{N}[X]$ such that, on every input $w$, the machine halts within $p(|w|)$ steps and returns the bit 1 if $w$ belongs to the language and 0 otherwise.

The definition uses mathlib's deterministic stack machines, the identity encoding of binary strings, and a singleton Boolean output. Time counts transitions of fixed finite instruction blocks. We also prove equivalence with elementary single-tape machines.

**Definition 2** (Finite-stack and elementary single-tape definitions of P)**.** The finite-stack class requires every stack alphabet to be finite. The single-tape class uses a deterministic machine whose transitions each move the head one square or write one symbol. Its alphabet and control states are finite. The input is written in its original order, starting at the head, with blank tape elsewhere. The two input symbols are distinct and different from blank. A halted control state determines the Boolean answer; work tape need not be erased.

**Definition 3** (Binary encoding of an input and a certificate)**.** To encode a pair $(x, y)$ of binary strings, replace each bit $b$ of $x$ by $0b$, then append a single 1 followed by $y$. This encoding has length $2|x|+|y|+1$ and has a unique decoding. In particular, a polynomial bound in the encoded length is a polynomial bound in the combined input and certificate lengths.

**Definition 4** (The complexity class NP)**.** A binary language $A$ belongs to NP if there are a verifier language $V \in$ P and a polynomial $p \in \mathbb{N}[X]$ such that $x \in A$ precisely when some binary certificate $y$ with $|y| \leq p(|x|)$ satisfies $\langle x, y\rangle \in V$. The verifier is a single deterministic polynomial-time decider on the explicit pair encoding, and its running time is polynomial in the combined input and certificate lengths. The certificate bound depends only on the original input length. This is the standard certificate definition of NP.

**Definition 5** (Finite Turing machines with a read-only input tape)**.** A machine has finitely many control states, a finite work alphabet with a blank symbol, a read-only binary input tape between distinct endmarkers, and one initially blank, semi-infinite work tape. Both heads start at position zero, which is the input's left endmarker and the work tape's leftmost cell. One transition reads the two scanned symbols, writes one work symbol, changes state, and moves each head by at most one cell. An outward move at a tape boundary leaves that head in place.

The finite transition table gives a finite set of possible actions for each state and pair of scanned symbols. A deterministic machine has at most one action in every such set. A configuration with no successor is terminal, and its control state supplies an accept/reject bit. Acceptance means that some computation branch reaches an accepting terminal configuration. A decider has a finite bound on the lengths of all branches for each input, and accepts exactly the strings in its language. This bound need not be computable or satisfy any specified time bound.

Only the work tape is charged as space. Bounding its head below $s$ at every reachable configuration bounds every visited cell to the prefix $0, \ldots, s-1$, including blank cells and cells later erased. The input head is confined to $|x|+2$ positions and cannot act as an unbounded free counter. The transition table sees neither head position nor the input length; it sees only the control state and scanned symbols.

**Definition 6** (Deterministic and nondeterministic space bounds)**.** For a function $s : \mathbb{N} \to \mathbb{N}$, the classes DSPACE and NSPACE below use the exact bound $s(n)$ on the number of work cells visited. There is one machine for all inputs, it halts on every branch, and every reachable configuration on an input of length $n$ respects the bound. DSPACE additionally requires deterministic transitions. Constant factors are quantified explicitly when the classical space classes are defined. We use $\lfloor \log_2(n+2) \rfloor$ as a positive logarithmic bound, so empty inputs are included without a special case.

**Definition 7** (The complexity class L)**.** A binary language belongs to L if a deterministic Turing machine decides membership using $O(\log n)$ work space and a separate read-only input tape. Precisely, some positive constant $c$ bounds work space by $c\lfloor \log_2(n+2) \rfloor$ on every input of length $n$.

**Definition 8** (The complexity class NL)**.** A binary language belongs to NL if a nondeterministic Turing machine decides membership using $O(\log n)$ work space and a separate read-only input tape. Every branch halts and respects the space bound; membership means that at least one branch accepts. The bound is $c\lfloor \log_2(n+2) \rfloor$ for one positive constant $c$.

**Definition 9** (The complexity class PSPACE)**.** A binary language belongs to PSPACE if one deterministic Turing machine decides membership using at most $p(n)$ work cells on inputs of length $n$, for some polynomial $p \in \mathbb{N}[X]$. The input tape is read-only and excluded from work space, and the machine always halts.

**Definition 10** (The complexity class NPSPACE)**.** A binary language belongs to NPSPACE if one nondeterministic Turing machine decides membership using at most $p(n)$ work cells on every branch on inputs of length $n$, for some polynomial $p \in \mathbb{N}[X]$. All branches halt; at least one accepts precisely when the input belongs to the language. The input tape is read-only and excluded from work space.

**Definition 11** (The complexity classes coNL and coNP)**.** For a class $\mathcal{C}$ of binary languages, co$\mathcal{C}$ consists of languages whose complements belong to $\mathcal{C}$. Complements are taken among all

finite binary strings. In particular, $A \in \mathrm{coNL}$ means $\overline{A} \in \mathrm{NL}$, and $A \in \mathrm{coNP}$ means $\overline{A} \in \mathrm{NP}$. This operation complements each language; it does not take the set-theoretic complement of the class of languages.

**Definition 12** (The complexity class EXPTIME)**.** A binary language belongs to EXPTIME if a deterministic Turing machine decides membership within $2^{p(n)}$ transitions on every input of length $n$, for some polynomial $p \in \mathbb{N}[X]$. The machine is the finite elementary single-tape machine defined here: input bits are distinct from blank, input appears in its original order, and each transition performs one move or one write. A terminal state's Boolean label gives the answer. The polynomial in the exponent may have any fixed degree; this is the usual EXPTIME, also called EXP.

## 2 Encoding properties

**Lemma 13.** *Decoding $\langle x, y\rangle$ returns exactly $(x, y)$.*

*Proof.* Induct on $x$. The empty word contributes only the delimiter; each additional bit contributes one two-bit block, which the decoder removes before applying the induction hypothesis. □

**Lemma 14.** *The function $(x, y) \mapsto \langle x, y\rangle$ is injective.*

*Proof.* Apply the decoder to an equality of encodings and use the preceding decoding identity. □

**Lemma 15.** *The encoded pair has length $|\langle x, y\rangle| = 2|x| + |y| + 1$.*

*Proof.* Induct on $x$. The delimiter contributes one bit, each bit of $x$ contributes two, and $y$ is appended unchanged. □

## 3 Polynomial-time machine models and complement

**Lemma 16** (Finite stack alphabets suffice)**.** *Requiring every work-stack alphabet to be finite leaves* P *unchanged.*

*Proof.* The input and output alphabets are finite. The machine has finite control, and the push instructions have finite ranges. Restrict each stack alphabet to these symbols. Every reachable transition and its time bound are preserved. Forgetting the finite-alphabet restriction gives the reverse inclusion. □

**Lemma 17** (Single-tape characterization of P)**.** *The elementary single-tape and stack-machine definitions of* P *coincide. Both simulations include polynomial bounds for input conversion, execution, and final output conversion.*

*Proof.* After restricting stack alphabets, compile the finitely many stacks to tape tracks with polynomial overhead. In the reverse direction, two stacks represent the portions of a tape on either side of its head, with linear overhead per simulated transition. The formal simulations include bounds for input conversion and final output conversion. □

**Lemma 18** (P is closed under complement)**.** *If a language of binary strings belongs to* P*, then its complement also belongs to* P*. The complement is taken in the set of all finite binary strings.*

*Proof.* Compose the output-alphabet equivalence with Boolean negation. The unchanged execution then returns the opposite Boolean answer with the same polynomial bound. Negating the correctness equivalence identifies the complementary language. □

**Lemma 19** (Single-tape P is closed under complement)**.** *The elementary single-tape class* P *is closed under complement.*

*Proof.* Transport complement closure across the proved equality of the single-tape and stack-machine classes. □

# 4 Inclusions

**Theorem 20** (The inclusion chain)**.** *The classes satisfy*

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} = \mathrm{NPSPACE} \subseteq \mathrm{EXPTIME}.$$

The following six statements prove each inclusion and the equality.

**Lemma 21.** $\mathrm{L} \subseteq \mathrm{NL}$.

*Proof.* Use the same machine and the same logarithmic space bound. Removing the requirement that each local observation has at most one action allows nondeterminism without changing any run. □

**Lemma 22.** $\mathrm{NL} \subseteq \mathrm{P}$.

*Proof.* Fix a nondeterministic decider using at most $c \log_2(n+2)$ work cells. Encode its state, input head, work head, and work-tape contents by a single integer. Encoding the tape in a fixed radix gives polynomially many candidate configurations, and every reachable configuration has an exact encoding. A finite program checks an edge by comparing the local transition and the finitely many possibly nonblank tape cells. Starting with the initial configuration marked, repeatedly mark every successor of a marked configuration. Each round scans polynomially many candidates; the proved configuration bound gives a polynomial number of rounds sufficient for every halting branch. An accepting terminal configuration is marked exactly when the original machine accepts. A checked compiler implements the bounded loops and table operations on a finite deterministic stack machine with polynomial running time, giving a characteristic function in the original definition of P. □

**Lemma 23.** $\mathrm{P} \subseteq \mathrm{NP}$.

*Proof.* Let $f$ be a polynomial-time characteristic function for $A$. On an encoded input-certificate pair, a fixed finite stack machine extracts the first component in linear time. It scans the two-bit blocks before the delimiter, clears the unused suffix, and restores the extracted bits to their original order. Compose this projection with the machine for $f$. Polynomial-time composition gives a verifier language in P. The zero polynomial is a certificate-length bound, and the empty certificate suffices. Decoding the first component proves the required equivalence with membership in $A$. □

**Theorem 24** (Savitch's theorem)**.** PSPACE = NPSPACE.

*Proof.* Every deterministic machine is also a nondeterministic machine with the same space bound. Conversely, suppose a nondeterministic decider uses at most $p(n)$ work cells. Enlarge the bound to the constructible function $s(n) = p(n) + n + 2$. The proved Savitch simulation gives a deterministic decider using at most $c\,s(n)^2$ work cells, which is a polynomial bound. □

**Lemma 25.** NP ⊆ PSPACE.

*Proof.* Let $p$ bound the certificate length, and fix a polynomial-time verifier. A finite nondeterministic machine constructs a unary supply of $2p(n) + 1$ symbols and guesses one bit for each symbol. Decoding the first component of the guessed word produces a certificate of length at most $p(n)$. Every permitted certificate occurs, since its encoding can be padded to exactly $2p(n)+1$ bits. The machine constructs the original input–certificate pair and simulates the verifier with finite control and a finite work alphabet. Its stack lengths are bounded by the encoded input length plus a constant times the verifier's running time. This is a polynomial in $n$, and every branch halts. Thus the language belongs to NPSPACE. Savitch's theorem (Theorem 24), proved above, gives PSPACE = NPSPACE and hence the conclusion. □

**Lemma 26.** NPSPACE ⊆ EXPTIME.

*Proof.* Apply the proved Savitch simulation to obtain a deterministic machine using polynomial work space. A halting computation cannot revisit a configuration, since the intervening transitions could then be repeated indefinitely. A machine with $q$ states, $g$ work symbols, and space bound $s$ on an input of length $n$ has at most $q(n+2)sg^s$ reachable configurations. Consequently its running time is exponential when $s$ is polynomial. The explicit stack simulation preserves each source transition as one stack-machine step and includes a linear bound for clearing the stacks and writing the answer. The stack-to-tape simulations have polynomial overhead in the input length and simulated time, so the final single-tape bound is still exponential. □

## 5 An open question

**Open question 27** (P versus NP)**.** The P versus NP problem asks whether every language with polynomially bounded, polynomial-time verifiable certificates can also be decided in deterministic polynomial time. We state the conjectured separation P ≠ NP as an open question.

This open question is not used by any formal proof in the submission.