RELATIONS BETWEEN COMPLEXITY CLASSES

A complexity class is specified by

- model of computation (multi-string TMs)
- mode of computation (deterministic, nondeterministic,. . .)
- resource (time, space, . . .)
- bound (function f)

A *complexity class* is the set of all *languages* decided by some multi-string Turing machine M operating in the appropriate mode, and such that, for any input x, M expends at most $f(|x|)$ units of the specified resource.

Reasonable bound functions

Definition. Let f be a function from nonnegative integers to nonnegative integers. Then f is a *proper complexity function* if f is nondecreasing and there is a k-string TM M_f with input and output such that on input x, $M_f(x) = \sqcap^{f(|x|)}$ and on any input x, M_f halts after $\mathcal{O}(|x| + f(|x|))$ steps and uses $\mathcal{O}(f(|x|))$ space besides its input.

- Examples of proper functions:
  $f(n) = c$, $f(n) = n$, $f(n) = \lceil \log n \rceil$,
  $\log^2 n$, $n \log n$, n^2, $n^3 + 3n$, 2^n, $\sqrt{n}$, $n!$
- If f and g are proper, so are, e.g., $f + g$, $f \cdot g$, 2^g.
- Only proper complexity functions are used as bounds.

Precise TMs

Definition. A TM (with input and output or not, deterministic or nondeterministic) is *precise* if there are functions f and g such that for every $n \geq 0$, for every input x of length n and for every computation of M,
(i) M halts after precisely $f(|x|)$ steps and
(ii) all of its strings (except for the first and last in case of a TM with input and output) are at halting of length precisely $g(|x|)$.

Precise TMs—cont'd

Proposition. Let a (deterministic or nondeterministic) TM M decide a language L within time (or space) $f(n)$, where f is a proper function. Then there is a precise TM M', which decides L in time (or space, respectively) $\mathcal{O}(f(n))$.

- M' obtained by first computing a yardstick/alarm clock $\sqcap^{f(|x|)}$ by M_f and then simulating M for exactly $f(|x|)$ steps/using exactly $f(|x|)$ units of space.

Complexity classes

- $\mathbf{TIME}(f)$, $\mathbf{NTIME}(f)$, $\mathbf{SPACE}(f)$, $\mathbf{NSPACE}(f)$

  where f is a proper complexity function.
- f can be a family of functions parametrized by an integer k; the complexity class is the union of all individual complexity classes.

Main complexity classes:

$$
\begin{array}{lcl}
\mathbf{P} & = & \mathbf{TIME}(n^k) = \bigcup_{j>0} \mathbf{TIME}(n^j) \\
\mathbf{NP} & = & \mathbf{NTIME}(n^k) = \bigcup_{j>0} \mathbf{NTIME}(n^j) \\
\mathbf{PSPACE} & = & \mathbf{SPACE}(n^k) \\
\mathbf{NPSPACE} & = & \mathbf{NSPACE}(n^k) \\
\mathbf{EXP} & = & \mathbf{TIME}(2^{n^k}) \\
\mathbf{L} & = & \mathbf{SPACE}(\log(n)) \\
\mathbf{NL} & = & \mathbf{NSPACE}(\log(n))
\end{array}
$$

Complements of nondeterministic classes

- Complement of a language $L \subseteq \Sigma^*$, $\overline{L} = \Sigma^* - L$.
- Complement of a decision problem A, A COMPLEMENT: answer "yes" iff for A answer is "no".

  Example. SAT COMPLEMENT: given a Boolean expression ϕ in CNF, is ϕ unsatisfiable?
- For any complexity class C, $\mathbf{co}C$ denotes the class $\{\overline{L} \mid L \in C\}$.
- All deterministic time and space complexity classes are closed under complement. Hence, e.g., $\mathbf{P} = \mathbf{coP}$
- Similarly for nondet. *space* complexity classes (to be shown).
- An open question: are nondeterministic *time* complexity classes closed under complement?

The hierarchy theorem

- A quantitative hierarchy result: with sufficiently greater time allocation, TMs are able to perform more complex computational tasks.
- Let $f(n) \geq n$ be a proper complexity function. Define

  $$H_f = \{M; x \mid M \text{ accepts input } x \text{ after at most } f(|x|) \text{ steps}\}$$

  (a time-bounded version of H, the HALTING language)

The hierarchy theorem—cont'd

Lemma. $H_f \in \mathbf{TIME}((f(n))^3)$

Proof sketch.

- A 4-string machine U_f deciding H_f in time $f(n)^3$ is based on
  (i) U universal TM
  (ii) single-string simulator of a multi-string machine
  (iii) the linear speedup machine and
  (iv) M_f computing the yardstick of length $f(n)$.
- First M_f computes the alarm clock $\sqcap^{f(|x|)}$ (string 4).
- Description of M is copied on string 3 and string 2 initialized to encode the initial state s and string 1 the input $\triangleright x$.

(Proof—cont'd)

- U_f simulates M and advances the alarm clock. If U_f finds M accepting the input within $f(|x|)$ steps, U_f accepts but if the alarm clock expires, U_f rejects.
- M is simulated by a single string: each step takes $\mathcal{O}(f(n)^2)$ time.
- Total running time $\mathcal{O}(f(n)^3)$ for $f(|x|)$ steps.
  By linear speedup the running time can be made at most $f(n)^3$.

The hierarchy theorem—cont'd

Lemma. $H_f \notin \mathbf{TIME}(f(\lfloor \frac{n}{2} \rfloor))$

Proof sketch.

- Assume that there is a TM M_{H_f} that decide H_f in time $f(\lfloor \frac{n}{2} \rfloor)$.
- Consider the machine
  $D_f(M)$: **if** $M_{H_f}(M; M) =$ "yes" **then** "no" **else** "yes".
  D_f on input M runs in the same time as M_{H_f} on $M; M$, i.e., in time $f(\lfloor \frac{2|M|+1}{2} \rfloor) = f(|M|)$.

(Proof—cont'd)

- If $D_f(D_f) =$ "yes", then $M_{H_f}(D_f; D_f) =$ "no", i.e., $D_f; D_f \notin H_f$. Hence, D_f fails to accept input D_f within $f(|D_f|)$ steps. As the running time of D_f on input D_f is bounded by $f(|D_f|)$ and it either accepts or rejects, D_f rejects input D_f, a contradiction.
- Hence, $D_f(D_f) \neq$ "yes". Then $D_f(D_f) =$ "no" and $M_{H_f}(D_f, D_f) =$ "yes". Thus, D_f accepts input D_f within $f(|D_f|)$ steps, i.e., $D_f(D_f) =$ "yes", a contradiction again.

Theorem. (The Time Hierarchy Theorem). If $f(n) \geq n$ is a proper complexity function, then the class $\mathbf{TIME}(f(n))$ is strictly contained within $\mathbf{TIME}((f(2n+1))^3)$

- $\mathbf{TIME}(f(n)) \subseteq \mathbf{TIME}((f(2n+1))^3)$ as f nondecreasing.
- By the first lemma: $H_{f(2n+1)} \in \mathbf{TIME}((f(2n+1))^3)$
- By the second lemma:
  $H_{f(2n+1)} \notin \mathbf{TIME}(f(\lfloor \frac{2n+1}{2} \rfloor)) = \mathbf{TIME}(f(n))$

Corollary. $\mathbf{P}$ is a proper subset of $\mathbf{EXP}$.

- $n^k = \mathcal{O}(2^n)$, hence, $\mathbf{P} \subseteq \mathbf{TIME}(2^n) \subseteq \mathbf{EXP}$.
- By the theorem:
  $\mathbf{TIME}(2^n) \subset \mathbf{TIME}((2^{2n+1})^3) \subseteq \mathbf{TIME}(2^{n^2}) \subseteq \mathbf{EXP}$

The hierarchy theorem—cont'd

Theorem. (The Space Hierarchy Theorem). If $f(n) \geq n$ is a proper complexity function, then the class $\mathbf{SPACE}(f(n))$ is a proper subset of $\mathbf{SPACE}(f(n) \log f(n))$.

- If non-proper complexity functions are allowed, counter-intuitive results are obtained:

Theorem. (The Gap Theorem). There is a recursive function f from the nonnegative integers to the nonnegative integers such that $\mathbf{TIME}(f(n)) = \mathbf{TIME}(2^{f(n)})$.

The reachability method

Theorem. Let $f(n)$ be a proper complexity function. Then

(a) $\mathbf{SPACE}(f(n)) \subseteq \mathbf{NSPACE}(f(n))$
  $\mathbf{TIME}(f(n)) \subseteq \mathbf{NTIME}(f(n))$

(b) $\mathbf{NTIME}(f(n)) \subseteq \mathbf{SPACE}(f(n))$

(c) $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{TIME}(c^{\log n + f(n)})$

- (a) A TM is a NTM, too.
- (b) Simulation of all choices within space $f(n)$.
- (c) By reachability method.

Consider (b) $\mathbf{NTIME}(f(n)) \subseteq \mathbf{SPACE}(f(n))$.

Proof.

- Let $L \in \mathbf{NTIME}(f(n))$. Hence there is a precise nondeterministic Turing machine N that decides L in time $f(n)$.
- Let d be the degree on nondeterminism (maximal number of possible moves for any state-symbol pair in Δ).
- Any computation of N is a $f(n)$-long sequence of nondeterministic choices.
  A sequence of $f(n)$ choices is a sequence of $f(n)$ integers from $0, 1, \ldots, d-1$.
- The simulating deterministic machine M considers all such sequences of choices and simulates N on each.

(Proof—cont'd)

- With sequence $(c_1, c_2, \ldots, c_{f(n)})$ M simulates the actions that N would have taken had N taken choice c_i at step i.
- If a sequence leads N to halting with "yes", then M does, too. Otherwise it considers the next sequence. If all sequences are exhausted without accepting, then M rejects.
- There is an exponential number of simulations to be tried but they can be carried out in *space* $f(n)$ by carrying them out one-by-one, always erasing the previous simulation to reuse space.
- As $f(n)$ is proper, the first sequence $0^{f(n)}$ can be generated in space $f(n)$.

Consider (c) $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{TIME}(c^{\log n + f(n)})$

Proof. (By *reachability method*)

- Consider a k-string NTM M with input and output deciding L within space $f(n)$. We develop a determ. method for simulating the nondeterm. computation of M on input x within time $c^{\log n + f(n)}$ where $n = |x|$ and c is a constant depending on M.
- Done by considering the *configuration graph* $G(M, x)$ of M: nodes are all possible configurations of M and there is an edge between two nodes (configurations) C_1 and C_2 iff $C_1 \xrightarrow{M} C_2$.
- $x \in L$ iff there is a path in $G(M, x)$ from $C_0 = (s, \triangleright, x, \triangleright, \epsilon, \ldots, \triangleright, \epsilon)$ to some configuration of the form $C =$ ("yes", $\ldots$).

(Proof—cont'd)

- A configuration $(q, w_1, u_1, \ldots, w_k, u_k)$ is a "snapshot" of a computation.
- For a machine with input and output deciding L, the output string is not interesting and for the input string only the position of the cursor can change.
  For all $k-2$ other strings the length is at most $f(n)$.
- A configuration can be represented as $(q, i, w_2, u_2, \ldots, w_{k-1}, u_{k-1})$ where $1 \leq i \leq n$ gives the cursor position on string 1.
- How many possible configurations does M have?
  At most $|K|(n+1)|\Sigma|^{(2k-2)f(n)} \leq nc_1^{f(n)} \leq c_1^{\log n + f(n)}$ for some constant c_1 depending on M.

(Proof—cont'd)

- Hence, deciding whether $x \in L$ holds can be done by solving the reachability problem for a graph with at most $c_1^{\log n + f(n)}$ nodes which can be performed with a quadratic algorithm in time $c_2 c_1^{2(\log n + f(n))} \leq c^{\log n + f(n)}$ with $c = c_2 c_1^2$.
- How to use a reachability algorithm:

  The graph $G(M, x)$ needs not to be represented explicitly.

  Instead, whenever the reachability algorithm needs to determine whether there is an edge from C to C', it examines the configurations C and C', the input x and determines whether C' can be yielded from C.

Class inclusions

Corollary. $\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$

Proof.

- $\mathbf{L} = \mathbf{SPACE}(\log(n)) \subseteq \mathbf{NSPACE}(\log n) = \mathbf{NL}$ by (a).
- $\mathbf{NL} = \mathbf{NSPACE}(\log n) \subseteq \mathbf{TIME}(c^{\log n + \log n}) = \mathbf{TIME}(n^{2\log c}) \subseteq \mathbf{P}$ by (c)
- By (a) $\mathbf{TIME}(n^k) \subseteq \mathbf{NTIME}(n^k)$ which implies $\mathbf{P} \subseteq \mathbf{NP}$.
- By (b) $\mathbf{NTIME}(n^k) \subseteq \mathbf{SPACE}(n^k)$ which implies $\mathbf{NP} \subseteq \mathbf{PSPACE}$.
- By (a) and (c) $\mathbf{SPACE}(n^k) \subseteq \mathbf{NSPACE}(n^k) \subseteq \mathbf{TIME}(c^{\log n + n^k}) \subseteq \mathbf{TIME}(2^{n^{k+c'}}) \subseteq \mathbf{EXP}$

Class inclusions—cont'd

By the space hierarchy theorem:

Corollary. $\mathbf{L}$ is a proper subset of $\mathbf{PSPACE}$.

$(\mathbf{L} = \mathbf{SPACE}(\log(n)) \subset \mathbf{SPACE}(\log(n)\log(\log(n))) \subseteq \mathbf{SPACE}(n^2) \subseteq \mathbf{PSPACE})$

- It is believed that *all* of the above inclusions of the complexity classes between $\mathbf{L}$ and $\mathbf{EXP}$ are proper but we only know that
- between $\mathbf{L}$ and $\mathbf{PSPACE}$ at least one of the inclusions is proper (but don't know which one) and
- that between $\mathbf{P}$ and $\mathbf{EXP}$ at least one of the inclusions is proper (but don't know which one).

Nondeterministic space

- How efficiently can we simulate nondeterministic space by deterministic space?
- By the previous theorem:
  $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{SPACE}(c^{\log n + f(n)})$
- Can we do better?
  Yes: nondeterministic space can be simulated with quadratic deterministic space (using Savitch's theorem)

Theorem. (**Savitch's theorem**)
REACHABILITY $\in \mathbf{SPACE}(\log^2 n)$.

Proof sketch.

- Given a graph G and nodes x, y and $i \geq 0$, define $PATH(x, y, i)$: there is a path from x to y of length at most 2^i.
- If G has n nodes, any simple path is at most n long and we can solve reachability in G if we can compute whether $PATH(x, y, \lceil \log n \rceil)$ holds for any given nodes of G.
- This can be done using *middle-first search*.

(Proof—cont'd)

- **function** $path(x, y, i)$ /* middle-first search */
  if $i = 0$ **then**
  if $x = y$ or there is an edge (x, y) in G **then** return "yes"
  else for all nodes z **do**
  if $path(x, z, i-1)$ and $path(z, y, i-1)$ **then** return "yes";
  return "no"
- If $i = 0$, then $path$ correctly determines $PATH(x, y, i)$.
  For $i > 0$, $path(x, y, i)$ returns "yes" iff there is a node z with $path(x, z, i-1)$ and $path(z, y, i-1)$ holding. By the inductive hypothesis there are paths from x to z and from z to y both at most 2^{i-1} long. Thus, there is a path from x to y at most 2^i long.

(Proof—cont'd)

- The algorithm is started with $path(x, y, \lceil \log n \rceil)$.
- $\mathcal{O}(\log^2 n)$ space bound can be realized by handling recursion using a stack containing a triple (x, y, i) for each recursive call active:
  for each node z put $(x, z, i-1)$ into the stack and call $path(x, z, i-1)$. If this fails, erase $(x, z, i-1)$ and put $(x, z', i-1)$ for the next z' otherwise erase $(x, z, i-1)$ and put $(z, y, i-1)$.
- As there are at most $\log n$ recursive calls active with each taking at most $3 \log n$ space, $\mathcal{O}(\log^2 n)$ space bound is achieved.

Corollary. $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{SPACE}((f(n))^2)$ for any proper complexity function $f(n) \geq \log n$.

Proof.

- To simulate an $f(n)$-space bounded NTM M on input x, run the previous algorithm on the configuration graph $G(M, x)$.
- The graph implicitly represented: the only time the input needs to be consulted is when determining whether there is an edge in $G(M, x)$ (done by examining the input and the machine M).
- The configuration graph has at most $c_1^{\log n + f(n)} \leq c^{f(n)}$ nodes. Hence the algorithm needs at most $(\log c^{f(n)})^2 = f(n)^2 \log^2 c = \mathcal{O}(f(n)^2)$ space (Savitch's theorem).

Corollary. $\mathbf{PSPACE} = \mathbf{NPSPACE}$

All nondeterministic space complexity classes are closed under complement

- A key underlying result (see below):
  the number of nodes reachable from x can be computed in nondeterministic $\log n$ space!
- The complement (the number of nodes not reachable) can be handled in nondeterministic $\log n$ space, too!

(Note that it is open (doubtful) whether nondeterministic *time* complexity classes are closed under complement.)

Functions computed by NTMs

- But when does a NTM M compute a function F from strings to strings?

  On input x, each computation of M either outputs the correct answer or "gives up" ending in "no" state. Furthermore, there needs to be at least one accepting computation.

- Such a machine observes a space bound $f(n)$ if for any input x, at halting all strings (except input and output) are of length at most $f(|x|)$.

Theorem. (The Immerman-Szelepscényi Theorem)
Given a graph G and a node x, the number of nodes reachable from x in G can be computed by a NTM within space $\log n$.

Proof. We are computing $|S(1)|, |S(2)|, \ldots, |S(n-1)|$ iteratively ($|S(i)|$ is computed from $|S(i-1)|$) where $S(k)$ is the set of nodes in G reachable from x by paths of length k or less.

$|S(n-1)|$: the number of nodes reachable from x in G, where n is the number of nodes in G.

(Proof—cont'd)

The nondeterministic algorithm:

$|S(0)| := 1$;
for $k := 1, 2, ..., n-1$ **do**
  $l := 0$;
  for each node $u := 1, 2, ..., n$ **do**
    check whether $u \in S(k)$, if so, $reply = true$;
    /* See below how this is implemented */
    if $reply = true$ **then** $l := l + 1$;
  end for;
  $|S(k)| := l$
end for

(Proof—cont'd) /* checking whether $u \in S(k)$ and setting reply */
$m := 0$; $reply1 := false$;
for each node $v := 1, 2, ..., n$ **do**
  /* check whether $v \in S(k-1)$ */
  $w_0 := x$;
  for $p := 1, 2, ..., k-1$ **do**
    guess a node w_p; **if** not $G(w_{p-1}, w_p)$ **then** "give up"
  end for
  if $w_{k-1} \neq v$ **then** "give up";
  $m := m + 1$; /* $v \in S(k-1)$ holds */
  if $G(v, u)$ **then** $reply1 := true$
end for
if $m < |S(k-1)|$ **then** "give up" **else** $reply := reply1$

(Proof—cont'd)

- The algorithm can be implemented on $\log n$-space bounded NTM as it only needs nine variables: $|S(k)|, k, l, u, m, v, p, w_p, w_{p-1}$ which each (an integer) can be stored in $\log n$ space.
- If $k = 0$, then $|S(k)| = 1$ as given by the algorithm.
- Assume that the algorithm computes correctly $|S(k-1)|$, $k > 0$.

(Proof—cont'd)

- We need to show that counter l is incremented iff $u \in S(k)$.

  l is incremented iff $reply = true$ iff computation does not "give up" and $m = |S(k-1)|$ (which is the correct value).

  Hence, if $reply = true$, then $u \in S(k)$ (there is a path constructed as $w_0, \ldots, w_{k-1}, u$).

  If $u \in S(k)$, then there is some $v \in S(k-1)$ such that $G(v, u)$. But as $m = |S(k-1)|$ all $v \in S(k-1)$ are verified as such and, thus, $reply = true$.

Nondeterministic space complexity classes are closed under complement

Corollary. If $f(n) \geq \log n$ is a proper complexity function, then $\mathbf{NSPACE}(f(n)) = \mathbf{coNSPACE}(f(n))$

Proof sketch.

- Suppose $L \in \mathbf{NSPACE}(f(n))$ is decided by an $f(n)$-space bounded NTM M. We construct an $f(n)$-space bounded NTM $\overline{M}$ deciding $\overline{L}$.
- On input x, $\overline{M}$ runs the previous algorithm on the configuration graph $G(M, x)$ where $G(M, x)$ is implicitly represented.

(Proof—cont'd)

- If while running the algorithm $\overline{M}$ finds an accepting configuration in any $S(k)$, then it halts and rejects.
- There is an upper bound $n_g = c^{f(n)}$ on the number of nodes in $G(M, x)$.
- Hence, if $|S(n_g - 1)|$ is computed without an accepting configuration, $\overline{M}$ can accept.
- $G(M, x)$ has at most $c^{f(n)}$ nodes. Hence, $\overline{M}$ needs at most $\log c^{f(n)} = \mathcal{O}(f(n))$ space.