Aalto University
School of Science

CS-E4530 Computational Complexity Theory

Lecture 6: Complexity Classes and their Relationships

Aalto University
School of Science
Department of Computer Science

Spring 2020

Topics

- Basic requirements for complexity classes
- Time and space complexity classes
- Hierarchy theorems
- The reachability method
- Class inclusions
- Simulating nondeterministic space
- Closure under complement

(C. Papadimitriou: *Computational Complexity*, Chapter 7)

7.1 Basic Requirements for Complexity Classes

A complexity class is specified by

- model of computation (multi-tape 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-tape 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

Not all functions provide reasonable bounds: some are too complicated to compute themselves within the bounds that they allow.

Definition (7.1)

A function $f : \mathbf{N} \to \mathbf{N}$ is a *proper complexity function* if f is nondecreasing and there is a k-tape TM M_f with input and output such that on any input x,

1. $M_f(x) = \sqcap^{f(|x|)}$ where $\sqcap$ is a *tally* ("quasi-blank") symbol,
2. M_f halts within $O(|x| + f(|x|))$ steps, and
3. M_f uses $O(f(|x|))$ space besides its input.

- Examples of proper complexity functions $f(n)$:

$$c,\ n,\ \lceil \log n \rceil,\ \log^2 n,\ n \log n,\ n^2,\ n^3 + 3n,\ 2^n,\ \sqrt{n},\ n!,\ \ldots$$

- If f and g are proper, so are, e.g., $f + g$, $f \cdot g$, 2^g.
- Only proper complexity functions will be used as bounds.

Precise Turing machines

Definition (7.2)

Let M be a deterministic/nondeterministic multi-tape Turing machine (with or without input and output).
Machine M 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,

1. M halts in precisely $f(|x|)$ steps and
2. all of its tapes (except those reserved for input and output whenever present) are at halting of length precisely $g(|x|)$.

Precise bounds will be convenient in various simulation results.

Simulating TMs with precise TMs

Proposition (7.1)

Let M be a deterministic or nondeterministic TM deciding a language L within time/space $f(n)$ where f is proper.
Then there is a precise TM M' which decides L in time/space $O(f(n))$.

Proof sketch

The simulating machine M' on input x:

1. computes a yardstick/alarm clock $\sqcap^{f(|x|)}$ using M_f (operating on a new set of tapes) and
2. using the yardstick (output tape of M_f)
   simulates M for exactly $f(|x|)$ steps *or*
   simulates M using exactly $f(|x|)$ units of space.

7.2 Time and Space Complexity Classes

- Given a proper complexity function f, we obtain following classes:
  $\mathbf{TIME}(f)$ (deterministic time)
  $\mathbf{NTIME}(f)$ (nondeterministic time)
  $\mathbf{SPACE}(f)$ (deterministic space)
  $\mathbf{NSPACE}(f)$ (nondeterministic space)
- The bound f can be a family of functions parameterised by a non-negative integer k; meaning the union of all individual classes.
- The most important are: $\mathbf{TIME}(n^k) = \bigcup_{j>0} \mathbf{TIME}(n^j)$
  $\mathbf{NTIME}(n^k) = \bigcup_{j>0} \mathbf{NTIME}(n^j)$

Key Complexity Classes

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

The relationships of these classes will be studied in the sequel.

Complements of decision problems

- Given an alphabet Σ and a language $L \subseteq \Sigma^*$, the *complement* of L is the language
$$\overline{L} = \Sigma^* - L.$$
- For a decision problem A, the answer for the problem"A COMPLEMENT" is "yes" iff the answer A is "no".

Example

SAT COMPLEMENT: Given a Boolean expression ϕ in CNF, is ϕ unsatisfiable?

Example

REACHABILITY COMPLEMENT: Given a graph (V,E) and vertices $v,u \in V$, is it the case that there is no path from v to u?

Closure under complement

- For any complexity class C, $\mathbf{co}C$ denotes the class

$$\{\overline{L} \mid L \in C\}.$$

  Example. As SAT $\in \mathbf{NP}$, then SAT COMPLEMENT $\in \mathbf{coNP}$.

- All deterministic time and space complexity classes C are *closed under complement*: if $L \in C$, then $\overline{L} \in C$.
  Proof. Exchange "yes" and "no" states of the deciding Turing machine.

- Hence, for example, $\mathbf{P} = \mathbf{coP}$.

- The same holds for nondeterministic *space* complexity classes (to be shown in the sequel).

- An important open question: are *nondeterministic time* complexity classes *closed under complement*?
  For instance, does $\mathbf{NP} = \mathbf{coNP}$ hold?

7.3 Hierarchy Theorems

- We derive a quantitative hierarchy result:
  with sufficiently greater time allocation, Turing machines are able to perform more complex computational tasks.
- For a proper complexity function $f(n) \geq n$, define

$$H_f = \{M;x \mid M \text{ accepts input } x \text{ within } f(|x|) \text{ steps}\}.$$

- Thus, H_f is the time-bounded version of H, i.e. the language of the HALTING problem.

Upper bound for H_f

Lemma (7.2)

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

Proof sketch

A 4-tape machine U_f deciding H_f in time $f(n)^3$ is based on

(i) the universal Turing machine U,

(ii) a machine M_f computing the yardstick of length $f(n)$ where n is the length of the input x,

(iii) the single-tape simulation of a multi-tape Turing machine, and

(iiv) the linear speedup trick.

Proof continued...

The machine U_f operates on input '$M;x$' as follows:

1. It copies the description of M to tape 3, initialises tape 2 to encode initial state s and initialises tape 1 to contain the input $\triangleright x$.
2. It uses M_f to compute the alarm clock $\sqcap^{f(|x|)}$ for M on tape 4.
3. Then U_f simulates M and advances the alarm clock at each simulated step. If U_f finds out that M accepts input x within $f(|x|)$ steps, then it accepts. But if the alarm clock expires, then U_f rejects.

Observations:

- Since the multi-tape machine M is simulated using a single tape, each simulation step takes $O(f(n)^2)$ time.
- The total running time thus is $O(f(n)^3)$ for $f(|x|)$ steps.
- The running time can be made at most $f(n)^3$ by treating several symbols as one, as in the proof of the linear speedup theorem.

Lower bound for H_f

Lemma (7.3)

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

Proof sketch

- Suppose there is a TM M_{H_f} that decides H_f in time $f(\lfloor \frac{n}{2} \rfloor)$.
- Consider $D_f(M)$: **if** $M_{H_f}(M;M) =$ "yes" **then** "no" **else** "yes".

  Thus, D_f on input M runs in time $f(\lfloor \frac{2|M|+1}{2} \rfloor) = f(|M|)$.
- If $D_f(D_f) =$ "yes", then D_f accepts input D_f within $f(|D_f|)$ steps, hence $D_f;D_f \in H_f$ and so $M_{H_f}(D_f;D_f) =$ "yes". But then $D_f(D_f) =$ "no", a contradiction.
- If $D_f(D_f) =$ "no", then D_f rejects input D_f within $f(|D_f|)$ steps, hence $D_f;D_f \notin H_f$ and so $M_{H_f}(D_f;D_f) =$ "no". But then $D_f(D_f) =$ "yes", a contradiction again.

The time hierarchy theorem

Theorem (7.4; Time Hierarchy Theorem)

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

Proof

- $\mathbf{TIME}(f(n)) \subseteq \mathbf{TIME}((f(2n+1))^3)$ as f is 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 (7.5)

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

Proof

- Since $n^k = O(2^n)$, we have $\mathbf{P} \subseteq \mathbf{TIME}(2^n) \subseteq \mathbf{EXP}$.
- It follows by the time hierarchy theorem that
  $\mathbf{TIME}(2^n) \subsetneq \mathbf{TIME}((2^{2n+1})^3) \subseteq \mathbf{TIME}(2^{n^2}) \subseteq \mathbf{EXP}$.

The space hierarchy theorem

Theorem (7.6; 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 $\mathbf{SPACE}(f(n) \log f(n))$.

However, counter-intuitive results are obtained if non-proper complexity functions are allowed.

Theorem (7.7: Gap Theorem)

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

Proof idea

The bound f can be defined so that no TM M computing on input x with $|x| = n$ halts in any number of steps between $f(n)$ and $2^{f(n)}$.

7.4 The Reachability Method

Theorem (7.8)

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

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

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

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

Proof

(a) Anyt TM is also an NTM.

(b) Simulation of all the choices within space $f(n)$ (see below).

(c) Proof by the reachability method (see below).

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

- Let $L \in \mathbf{NTIME}(f(n))$. Hence, there is a precise nondeterministic Turing machine N that decides L in time $f(n)$.
- We show how to construct a deterministic machine M that simulates N within the space bound $f(n)$.
- Let d be the degree on nondeterminism of N (maximal number of possible moves for any state-symbol pair in Δ).
- Any computation of N on input x corresponds to an $f(n)$-long sequence of nondeterministic choices (represented by integers $0, 1, \ldots, d-1$) where $n = |x|$.
- 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 are an exponential number of simulations to be carried out, but they can all be completed in *space* $f(n)$ by running them one-by-one, and always erasing the previous simulation to reuse space.
- As $f(n)$ is proper, the first sequence $0^{f(n)}$ can be generated in space $f(n)$.

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

The *reachability method* is used to prove the claim.

- Consider a k-tape *nondeterministic* TM N with input and output which decides a language L within space $f(n)$.
- We develop a deterministic method for simulating the nondeterministic computation of N on input x within time $c^{\log n + f(n)}$ where $n = |x|$ and c is a constant depending on N.
- The *configuration graph* $G(N,x)$ of N with input x is used: the vertices of the graph are all possible configurations of N with input x, and there is an edge between two vertices (configurations) C_1 and C_2 iff $C_1 \vdash_N C_2$.
- Now $x \in L$ iff there is a path from $C_0 = (s, \triangleright, x, \triangleright, \varepsilon, \ldots, \triangleright, \varepsilon)$ to some configuration of the form $C = (\text{“yes”}, \ldots)$ in $G(N,x)$.

Proof—cont'd.

- A configuration $(q, w_1, u_1, \ldots, w_k, u_k)$ is a complete "snapshot" of a computation.
- Since N is a machine with input and output *deciding* L with the bound $f(n)$, we observe that for a configuration:
  – the output tape can be neglected,
  – for the input tape, only the cursor position can change, and
  – for all other $k-2$ tapes, the length is at most $f(n)$.
- A configuration can thus 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 the input tape.
- How many possible configurations does N have? At most

$$|K|(n+1)(|\Sigma|^{f(n)})^{2(k-2)}$$

$$\leq |K|2n(|\Sigma|^{2(k-2)})^{f(n)} \leq nc_1^{f(n)} \leq c_1^{\log n + f(n)}$$

  for some constant $c_1 \geq 2$ depending on N.

Proof—cont'd.

- Hence, deciding whether $x \in L$ holds can be done by solving a reachability problem for a graph with at most $c_1^{\log n + f(n)}$ vertices.
- The problem can be solved, say, with a quadratic algorithm in time $c_2 c_1^{2(\log n + f(n))} \leq c^{\log n + f(n)}$ with $c = c_2 c_1^2$.
- The graph $G(N, x)$ does not need to be represented explicitly (e.g., as an adjacency matrix) for the reachability algorithm.
- Given the machine N, the existence of an edge from C to C' can be determined on the fly by examining C, C', and the input x.

7.5 Class Inclusions

Corollary (7.9)

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

Proof

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

Which inclusions are proper?

Corollary (7.10)

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

Proof

By the space hierarchy theorem, $\mathbf{L} = \mathbf{SPACE}(\log(n)) \subsetneq \mathbf{SPACE}(\log(n)\log(\log(n))) \subseteq \mathbf{SPACE}(n^2) \subseteq \mathbf{PSPACE}$.

It is believed that *all* inclusions of the complexity classes in $\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$ are proper.

However, we only know that

- at least one of the inclusions between $\mathbf{L}$ and $\mathbf{PSPACE}$ is proper (but don't know which) and
- at least one of the inclusions between $\mathbf{P}$ and $\mathbf{EXP}$ is proper (but don't know which).

7.6 Simulating Nondeterministic Space

- How efficiently can we simulate nondeterministic space by deterministic space?
- It follows by the previous theorem that

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

  But can we do better than this?
- *Yes!* In fact, nondeterministic space can be simulated in deterministic space with only a quadratic overhead.

Savitch's theorem

Theorem (7.11)

REACHABILITY $\in \mathbf{SPACE}(\log^2 n)$.

Proof sketch

- Given a graph G and vertices 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 vertices, any simple path is at most n edges long and we can solve reachability in G if we can compute whether $PATH(x, y, \lceil \log n \rceil)$ holds for any given vertices x, y of G.
- This can be done using *middle-first search* within space bound $\log^2 n$.

Proof continued

- **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 vertices z **do**
  if $path(x,z,i-1)$ and $path(z,y,i-1)$ **then** return "yes";
  return "no"
- We prove that $path(x,y,i)$ correctly determines $PATH(x,y,i)$ by induction on $i = 0,1,2,\ldots$:
  If $i = 0$, then clearly $path$ correctly determines $PATH(x,y,0)$.
  For $i > 0$, $path(x,y,i)$ returns "yes" iff there is a vertex z satisfying $path(x,z,i-1)$ and $path(z,y,i-1)$. By the induction hypothesis, there are then paths from x to z and from z to y both of length at most 2^{i-1}. Thus there is a path from x to y of length at most 2^i.

Proof continued

- The algorithm is started with $path(x, y, \lceil \log n \rceil)$.
- The $O(\log^2 n)$ space bound can be achieved by managing recursion using a stack that contains a triple (x, y, i) for each active recursive call $path(x, y, i)$. The stack, with triple (x, y, i) at the top, is handled as follows:
  - Generate all vertices z one after the other reusing space.
  - For each z, push $(x, z, i-1)$ on the stack and call $path(x, z, i-1)$.
  - If this returns "no", erase $(x, z, i-1)$ and move to the next z.
  - If a "yes" answer is obtained, then erase $(x, z, i-1)$, push $(z, y, i-1)$ on the stack and call $path(z, y, i-1)$.
  - If this returns "no", erase $(z, y, i-1)$ and move to the next z. Otherwise return "yes" on $path(x, y, i)$.
- At any moment, there are at most $\log n$ active recursive calls, each taking at most $3 \log n$ space on the stack. The $O(\log^2 n)$ space bound follows.

(Interestingly, REACHABILITY for *undirected graphs* is solvable even in $\mathbf{SPACE}(\log n)$ [O. Reingold 2004].)

Corollary (7.12; Savitch's Theorem, 1970)

For any proper complexity function $f(n) \geq \log n$,

$$\mathbf{NSPACE}(f(n)) \subseteq \mathbf{SPACE}((f(n))^2).$$

Proof

- To simulate an $f(n)$-space bounded NTM N on input x, run the previous algorithm on the configuration graph $G(N,x)$.
- The edges of the graph $G(N,x)$ are determined on the fly by examining the input x.
- The configuration graph has at most $c_1^{\log n + f(n)} \leq c^{f(n)}$ vertices for some c.
- By Savitch's theorem, the algorithm needs at most $(\log c^{f(n)})^2 = f(n)^2 \log^2 c = O(f(n)^2)$ space.

Corollary (7.13)

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

☞ Nondeterminism is less powerful for space than for time. (Maybe.)

7.7 Closure under Complement

- A key result about reachability will be established:
  the number of vertices reachable from a vertex x can be computed in nondeterministic $\log n$ space.
- The complement (the number of vertices *not* reachable from x) can be handled in nondeterministic $\log n$ space, too.
  (This quantity can be obtained by a simple subtraction.)
- This implies that *nondeterministic space is closed under complement*.
- It is open (and doubtful) whether nondeterministic *time* complexity classes are closed under complement.

Functions computed by NTMs

When does an NTM M compute a function F from strings to strings?

- On input x, each computation of M either
  – outputs the correct answer $F(x)$ or
  – enters the rejecting "no" state.
- At least one computation must end up with $F(x)$, which must be unique for all such computations.
- Such a machine observes a space bound $f(n)$ iff for any input x, at halting all tapes (except the ones reserved for input and output) are of length at most $f(|x|)$.

Immerman-Szelepcsényi theorem

Theorem (7.14; Immerman-Szelepcsényi theorem, 1987)

Given a graph G and a vertex x, the number of vertices reachable from x in G can be computed by an NTM within space $\log n$.

Proof

- Let us define $S(k)$ as the set of vertices in G which are reachable from x via paths of length k or less.
- The strategy is to compute values $|S(1)|, |S(2)|, \ldots, |S(n-1)|$ iteratively and recursively, i.e. $|S(i)|$ is computed from $|S(i-1)|$.
- Given that the number of vertices in G is n, the number of vertices reachable from x in G is $|S(n-1)|$.
- Let $G(v,u)$ mean that $v = u$ or there is an edge from v to u in G.

Proof continued

The nondeterministic algorithm:

$|S(0)| \leftarrow 1;$
for $k \leftarrow 1, 2, \ldots, n-1$ **do**
    $l \leftarrow 0;$
    for each vertex $u \leftarrow 1, 2, \ldots, n$ **do**
        check whether $u \in S(k)$ and set *reply* accordingly;
        /* See below how this is implemented */
        if $reply = true$ **then** $l \leftarrow l + 1;$
    end for;
    $|S(k)| \leftarrow l$
end for

Proof continued

/* Check whether $u \in S(k)$ and set *reply* */

$m \leftarrow 0$; $reply \leftarrow false$;

/* $m \sim$ count of vertices in $S(k-1)$ reached in a nondet. trial */

/* $reply \sim$ was $u \in S(k)$ discovered? */

for each vertex $v \leftarrow 1, 2, \ldots, n$ **do**

/* check whether $v \in S(k-1)$ */

$w_0 \leftarrow x$; $path \leftarrow true$

for $p \leftarrow 1, 2, \ldots, k-1$ **do**

guess a vertex w_p; **if** not $G(w_{p-1}, w_p)$ **then** $path \leftarrow false$

end for

if $path = true$ and $w_{k-1} = v$ **then**

$m \leftarrow m+1$; /* $v \in S(k-1)$ holds */

if $G(v, u)$ **then** $reply \leftarrow true$

end if

end for

if $m < |S(k-1)|$ **then** "give up" (end in "no" state)

Proof continued

- Note that only $\log n$-space is needed as there are only nine variables: $|S(k-1)|, k, l, u, m, v, p, w_p, w_{p-1}$ which each (an integer) can be stored in $\log n$ space.
- The algorithm computes correctly $|S(k)|$ (by induction on k):
  – If $k = 0$, then $|S(k)| = 1$ as given by the algorithm.
  – For $k > 0$, consider a computation that does not "give up". We need to show that counter l is incremented iff $u \in S(k)$.
  If counter l is incremented, then $\mathit{reply} = \mathit{true}$ implying that $u \in S(k)$, i.e. there is a path $(x =)w_0, \ldots, w_{k-1}(= v), u$.
  If $u \in S(k)$, then there is some $v \in S(k-1)$ such that $G(v, u)$. But as the computation does not "give up", $m = |S(k-1)|$ (which is the correct value by induction) and therefore all $v \in S(k-1)$ are verified as such and, thus, reply is set to true.
  – Moreover, clearly there is at least one accepting computation where paths to the members of $S(k-1)$ are correctly guessed.

Closure under Complement

Corollary (7.15)

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 N. We build an $f(n)$-space bounded NTM $\overline{N}$ deciding $\overline{L}$.
- On input x, $\overline{N}$ runs the previous algorithm on the configuration graph $G(N,x)$ associated with N and x.
- $\overline{N}$ rejects if it finds an accepting configuration in any $S(k)$.
- Since $G(N,x)$ has at most $n_g = c^{f(n)}$ vertices, then $\overline{N}$ can accept if $|S(n_g - 1)|$ is computed without an accepting configuration.
- Due to bound n_g, $\overline{N}$ needs at most $\log c^{f(n)} = O(f(n))$ space.

Learning Objectives

- The definitions and background of major complexity classes: $\mathbf{P}$, $\mathbf{NP}$, $\mathbf{PSPACE}$, $\mathbf{NPSPACE}$, $\mathbf{EXP}$, $\mathbf{L}$, and $\mathbf{NL}$.
- The knowledge of basic relationships between complexity classes (inclusions and proper inclusions).
- Savitch's theorem and Immerman-Szelepscényi theorem.