2. Time and Space Complexity Classes

Goal:

- Introduce basic complexity classes
- The classes have proven useful because
  - ↳ they characterize important problems (computing, searching/guessing, playing against an opponent)
  - ↳ they are robust under reasonable changes to the model
    (P is the same class of problems
    no matter whether we take
    - polynomial-time Turing maschines
    - polynomial-time while programs
    - polynomial-time RAM maschines
    - polynomial-time C++ programs)

2.1 Recapitulation: Turing Maschines

Definition

A Turing maschine (or semi-decider) M is a tupel

$$M = (Q, \Sigma, \Gamma, q_0, \delta, \{q_{acc}, q_{rej}\}),$$

where Q, Σ, Γ, q_0 and δ (deterministic or non-deterministic) are defined as in chapter 14.

The set of states Q contains in particular

- the accepting state q_{acc} and
- the rejecting state $q_{rej} \neq q_{acc}$

Further requirement: Once the maschine reaches q_{acc}/q_{rej}, it no longer changes anything

$$\forall b \in \Gamma \quad \delta(q_{acc}, b) = (q_{acc}, b, N) \text{ and } \delta(q_{rej}, b) = (q_{rej}, b, N)$$

We also call q_{acc} and q_{rej} halting states
and say that the maschine halts if q_{acc} or q_{rej} is entered.

The configurations and
the transition relation $\rightarrow$ between configuration are defined as in chapter 14.

We call a configuration of shape $u\, q_{acc}\, v$ accepting
and a configuation of shape $u\, q_{rej}\, v$ rejecting.
Accepting and rejecting configuations are halting configuations

As in chapter 14, the language of M is

$$L(M) = \{ w \in \Sigma^* \mid q_0\, w \longrightarrow^* u\, q_{acc}\, v \}$$

When a TM M is started on input $w \in \Sigma^*$,
there are four possible outcomes.

- M may accept $\Big\}$ halt
- M may reject
- M may loop (not halt)
- M may get stuck, i.e. there is no appropriate transition)
  (can not happen in q_{acc}/q_{rej} or if M is deterministic)

Distinguishing looping from taking a long time
(to accept / reject / get stuck) is difficult.

We are interested in maschines that
- halt on every input and
- halt in every computation (for NTMs)

Such maschines are said to be total or deciders.

Theorem (Recapitulation)
For every (total) NTM M, there is a total DTM M'
with $L(M) = L(M')$

2.2 Time Complexity

Goal: Define $\overset{D}{\underset{N}{}}TIME_{(k)}(t(n))$.

Let M be a Turing machine (potentially nondeterministic, potentially several tapes).

Let $x \in \Sigma^*$ be an input of M.

- We define

$$Time_M(x) := \max\{\text{number of transitions on path } p \mid p \text{ a computation path of } M \text{ on } x\}.$$

If M does not halt (on some path), we set $Time_M(x) := \infty$.

Note that for a deterministic Turing machine,
there is precisely one computation path.

- For $n \in \mathbb{N}$, we define the <u>time complexity of M</u> as

$$Time_M(n) := \max\{Time_M(x) \mid |x| = n\}$$

$Time_M(n)$ measures the <u>worst case</u> behavior of M on inputs of length n.

- Let $t : \mathbb{N} \to \mathbb{N}$ be some function.
We say that M is <u>t-time-bounded</u> (also written <u>$t(n)$-time-bounded</u>),
if $Time_M(n) \leq t(n)$ for all $n \in \mathbb{N}$.

<u>Definition</u>:

Let $t : \mathbb{N} \to \mathbb{N}$. Then

$$DTIME_k(t(n)) := \{L(M) \mid M \text{ is a } k\text{-tape DTM that is a decider and } t(n)\text{-time-bounded}\}.$$

$$NTIME_k(t(n)) := \{L(M) \mid M \text{ is a } k\text{-tape NTM that is a decider and } t(n)\text{-time-bounded}\}.$$

We write DTIME($t(n)$) and NTIME($t(n)$)
if we assume the Turing machine to be 1-tape.

Note:
Sublinear time is not meaningful for Turing machines
(that do not have random access to the input).
To render this formally, let M be a DTM
and assume there is an $n \in \mathbb{N}$ so that
M reads at most $n-1$ symbols of the input x,
for every x with $|x| = n$.
Then there are words $a_1, \dots, a_m$ with $|a_i| < n$ for all $1 \leq i \leq m$
so that

$$L(M) = \bigcup_{i=1}^{m} a_i \Sigma^*.$$

2.3 Space Complexity

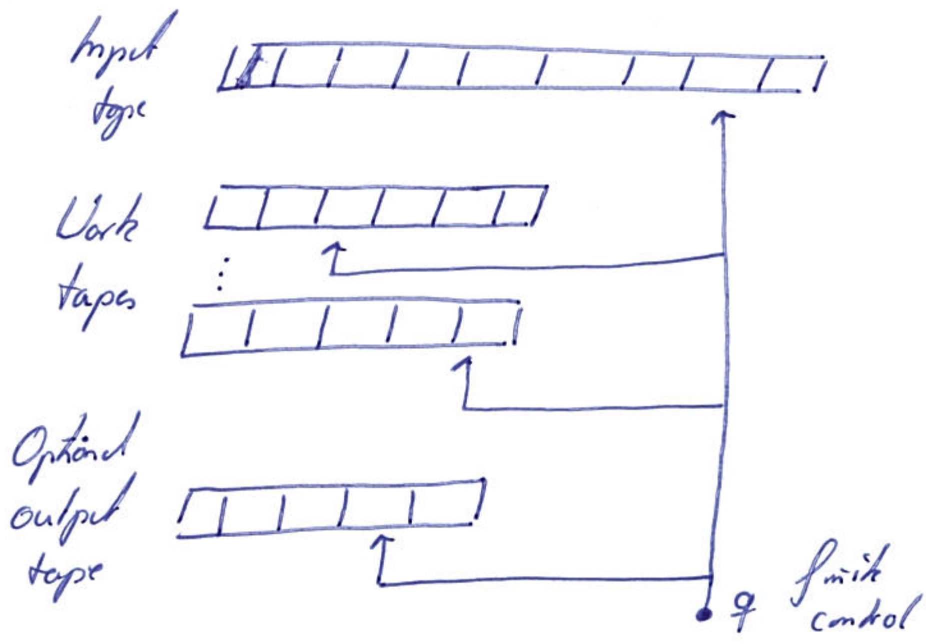
Goal: Define $\genfrac{}{}{0pt}{}{D}{N}$SPACE$_{(k)}$ ($s(n)$).

Assumption:
- Different from the case of time complexity, it is interesting to study computations that run in sublinear space.
- Therefore, we will assume that a Turing machine has an extra input tape.

The input tape is
- read-only and
- not counted towards the space consumption.

(Technically, read-only amounts to requiring that whenever the Turing machine reads a symbol, it has to write the same symbol).

Graphically:



- Let M be a Turing machine (with separate input tape, potentially several work tapes, potentially nondeterministic).
  Let $x \in \Sigma^*$ be an input of M and
  let c be a configuration of M.
  Then

$$\text{Space}(c) := \max \{ |w| \mid w \text{ represents the content (in } c\text{) of one of the } \underline{\text{work tapes}} \}.$$

- We define

$$\text{Space}_M(x) := \max \{ \text{Space}(c) \mid c \text{ a configuration that occurs in a computation of } M \text{ on } x \}$$

  If the space grows unboundedly, we set $\text{Space}_M(x) := \infty$.

- For $n \in \mathbb{N}$, the <u>space complexity</u> of M is

$$\text{Space}_M(n) := \max \{ \text{Space}_M(x) \mid |x| = n \}.$$

- Let $s : \mathbb{N} \to \mathbb{N}$ be some function.
  We say that M is <u>s-space-bounded</u>,
  if $\text{Space}_M(n) \leq s(n)$ for all $n \in \mathbb{N}$.

Definition:

Let $s: \mathbb{N} \to \mathbb{N}$. Then

$DSPACE_k(s(n)) := \{ L(M) \mid M$ is a k-tape DTM (with an extra input tape) that is a decider and $s(n)$-space-bounded $\}$

$NSPACE_k(s(n)) := \{ L(M) \mid M$ is a k-tape NTM (with an extra input tape) that is a decider and $s(n)$-space-bounded $\}$

Example:

Consider the language

$L = \{ x \in \{a, b\}^* \mid$ the number of as in x equals the number of bs in $x \}$.

We show that $L \in SPACE(O(\log n))$.

We read the input from left to right.
On the work tape, we keep a binary counter.
- If we read an a, we increment (+1) the binary counter.
  If we read a b, we decrement (−1) the binary counter.
  We accept, if the counter value reached in the end is 0.
- In every step, we store a number $\leq |x|$ in binary on the work tape.
  This needs $\log |x|$ bits.
- The construction requires us to increment and decrement in binary.
  This does not cause space overhead.

2.4 Common Complexity Classes

Definition:

$$L := DSPACE(\mathcal{O}(\log n)) \qquad \text{(aka LOGSPACE)}$$

$$NL := NSPACE(\mathcal{O}(\log n)) \qquad \text{(aka NLOGSPACE)}$$

$$P := \bigcup_{k \in \mathbb{N}} DTIME(\mathcal{O}(n^k)) \qquad \text{(aka PTIME)}$$

$$NP := \bigcup_{k \in \mathbb{N}} NTIME(\mathcal{O}(n^k))$$

$$PSPACE := \bigcup_{k \in \mathbb{N}} DSPACE(\mathcal{O}(n^k))$$

$$NPSPACE := \bigcup_{k \in \mathbb{N}} NSPACE(\mathcal{O}(n^k))$$

$$EXP := \bigcup_{k \in \mathbb{N}} DTIME(2^{\mathcal{O}(n^k)}) \qquad \text{(aka EXPTIME)}$$

$$NEXP := \bigcup_{k \in \mathbb{N}} NTIME(2^{\mathcal{O}(n^k)}) \qquad \text{(aka NEXPTIME)}$$

$$EXPSPACE := \bigcup_{k \in \mathbb{N}} DSPACE(2^{\mathcal{O}(n^k)})$$

$$NEXPSPACE := \bigcup_{k \in \mathbb{N}} NSPACE(2^{\mathcal{O}(n^k)}).$$

Definition:

Let $C \subseteq \mathcal{P}(\{0,1\}^*)$ be a complexity class.

We define

$$co\text{-}C := \{ L \subseteq \{0,1\}^* \mid \bar{L} \in C \text{ with } \bar{L} := \{0,1\}^* \setminus L \}$$

to be the complement class of C.

Note that co-C is not the complement of C, but the complements of the sets in C.

Intuitively, a problem in co-C contains the "no"-instances of a problem in C.

Example:

UNSAT := { φ a formula in CNF | φ is not satisfiable }.

Then

UNSAT $\in$ co-NP, because

$$\overline{\text{UNSAT}} = \text{SAT} \in \text{NP}.$$

Roughly, the goal of the lecture is to

(1) understand the aforementioned complexity classes (what are the problems they capture, what do their algorithms look like).

(2) understand the relationships among the classes.

A simple theorem of form (2) is the following.

Theorem:

If C is a deterministic time or space complexity class, then C = co-C.

For example, L = co-L, P = co-P, PSPACE = co-PSPACE.

Definition:

A complexity class C is said to be closed under complement, if for all $L \in C$ we have $\overline{L} \in C$.

Claim:

C = co$C$ iff C is closed under complement iff co-C is closed under complement.

Further basic inclusions of the form (2) are the following:

<u>Lemma:</u>

Let $t, s : \mathbb{N} \to \mathbb{N}$.

$$DTIME(t(n)) \subseteq NTIME(t(n))$$
$$DSPACE(s(n)) \subseteq DSPACE(s(n))$$
$$DTIME(t(n)) \subseteq DSPACE(t(n))$$
$$NTIME(t(n)) \subseteq NSPACE(t(n)).$$

<u>Proof:</u>

- For the former two inclusions, we note that every DTM is also an NTM.
- For the latter two inclusions, we note that a machine can only scan one cell per step. So the tape usage is bounded by the time.