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 machines
• polynomial-time while programs
• polynomial-time RAM machines
• polynomial-time C++ programs)

2.1 Turing Machines

Goal: Define Turing machine (deterministic, non-deterministic, multi-tape) as recognizers and deciders of languages.

Definition
A deterministic 1-tape TM is a 9-tuple

$$M = (Q, \Sigma, \Gamma, \#, \sqcup, \delta, q_0, q_{accept}, q_{reject})$$

where
- Q is a finite set of states with initial state q_0, accepting state q_{accept} and rejecting state q_{reject}
- Σ the input alphabet (finite) not containing $\sqcup$ and $\#$

- Γ is the tape alphabet with $\Sigma \subseteq \Gamma$,
  $\sqcup \in \Gamma$ the blank symbol
  $\triangleright \in \Gamma$ the left endmarker
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ the transition function.

Further requirements:

Endmarker is never overwritten:

$$\forall p \in Q \; \exists q \in Q : \; \delta(p, \triangleright) = (q, \triangleright, R)$$

Once the machine halts, it no longer changes anything:

$$\forall b \in \Gamma \; \exists d, d' : \; \delta(q_{accept}, b) = (q_{accept}, b, d)$$
$$\delta(q_{reject}, b) = (q_{reject}, b, d').$$

For the semantics of a Turing machine,
define a notion of · configuration and
· a transition relation among configurations.

Definition:

Let $M = (Q, \Sigma, \Gamma, \triangleright, \sqcup, \delta, q_0, q_{accept}, q_{reject})$.

- A configuration of M is a triple
  $$u q v \in \Gamma^* \times Q \times \Gamma^*$$
  with the idea that the head is on the first symbol of v.
- The transition relation $\to \subseteq (\Gamma^* \times Q \times \Gamma^*) \times (\Gamma^* \times Q \times \Gamma^*)$
  is defined by
  $$u a q b v \to u q' a c v, \text{ if } \delta(q, b) = (q', c, L)$$
  $$u a q b v \to u a c q' v, \text{ if } \delta(q, b) = (q', c, R).$$
- A configuration uq is understood to be equivalent to $uq\sqcup$.

- The initial configuration of M on input $w \in \Sigma^*$ is
$$q_0 \sqcup w.$$
- If configuration of the form $u\, q_{accept}\, v$ is accepting.
If configuration of the form $u\, q_{reject}\, v$ is rejecting.
Accepting and rejecting configurations are halting configurations.
- M accepts input w, if there is a sequence of configurations
$$C_1 \to C_2 \to \dots \to C_n$$
where • C_1 is the initial configuration of M on input w
• C_n is an accepting configuration.

The language of M is
$$L(M) := \{ w \in \Sigma^* \mid M \text{ accepts } w \}.$$

Definition:
A language $L \subseteq \Sigma^*$ is called (Turing) recognizable (also called recursively enumerable), if
$$L = L(M) \quad \text{for some TM } M.$$

When a TM is started on an input, there are three possible outcomes
- M may accept
- M may reject
- M may loop (not halt).

So M may fail to accept by rejecting or by looping.

Distinguishing looping from taking a long time (to accept or reject) is difficult.

- We are only interested in machines that halt on all inputs, i.e., the never loop.
  Such machines are said to be total, also called deciders.

Definition:
A language $L \subseteq \Sigma^*$ is called (Turing) decidable (also called recursive), if

$$L = L(M) \text{ for some total TM } M.$$

A multi-tape TM is like an ordinary TM but with several tapes.
Initially, the input is on tape 1, the remaining tapes are empty.
The transition function allows for reading, writing, and moving the heads on some (or all) tapes simultaneously:

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$$

Theorem:
For every multi-tape TM M there is a 1-tape TM M' with $L(M) = L(M')$.
Hence, a language is recognizable if and only if some multi-tape TM recognizes it.

A nondeterministic TM (NTM) may, at any point in a computation proceed according to several possibilities.
Formally, the transition function takes the form

$$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\}).$$

The computation of a nondeterministic TM is a tree whose branches correspond to different possibilities of the machine.
If some branch of the tree leads to an accepting configuration, the machine accepts the input.

$$L(M) := \{ w \in \Sigma^* \mid \text{some branch of } M \text{ started on } w \text{ accepts} \}$$

M is called total or a decider.
if for every input $w \in \Sigma^*$ all branches halt.

Theorem:

- For every (total) NTM M there is a (total) DTM M' with $L(M) = L(M')$.
- Hence, a language is recognizable if and only if some NTM recognizes it.
- Furthermore, a language is decidable if and only if some NTM decides it.

Recapitulation (Decidable and Semidecidable):

- A property $P: \Sigma^* \to \mathbb{B}$ is called decidable, if $\{ x \in \Sigma^* \mid P(x) = \text{true} \}$ is a recursive set.
  (there is a total TM that accepts the input strings having property P, and rejects the strings that violate P)
- A property P is semidecidable,

If $\{x \in \Sigma^* \mid P(x) = \text{true}\}$ is a recursively enumerable set.

In short:

- recursive and recursively enumerable apply to sets
- decidable and semidecidable apply to properties.

The notions are interchangeable:

P decidable $\Leftrightarrow$ $\{x \mid P(x)\}$ recursive

A recursive $\Leftrightarrow$ "$x \in A$" decidable.

P semidecidable $\Leftrightarrow$ $\{x \mid P(x)\}$ recursively enumerable

A recursively enumerable $\Leftrightarrow$ "$x \in A$" semidecidable.

2.2 Time Complexity

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

Let M be a Turing machine (potentially nondeterministic, potentially several tapes).
Let $x \in \Sigma^*$ be an input of M.

- We define
  $Time_M(x) := \max\{$ number of transitions on path p | p a computation path of M 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

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

$NTIME_k(t(n)) := \{L(M) \mid M$ is a k-tape NTM that is a decider and $t(n)$-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 $n_0 \in \mathbb{N}$ 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 $1 \le i \le m$
so that

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

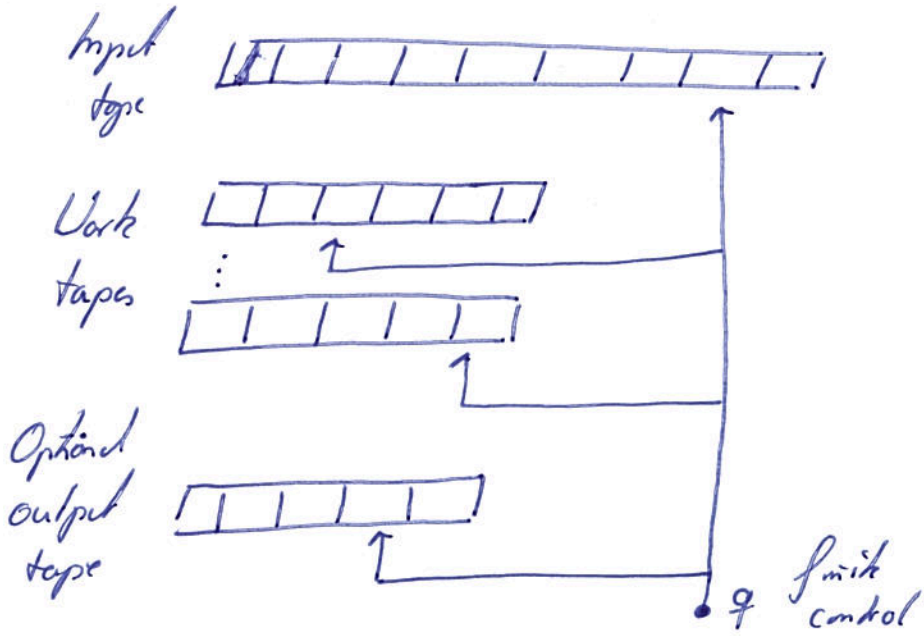
2.3 Space Complexity

Goal: Define $\mathrm{NSPACE}(L(n))$ $(s(n))$.

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

- 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 space complexity 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 s-space-bounded,
if $\text{Space}_M(n) \leq s(n)$ for all $n \in \mathbb{N}$.

Definition:

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

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

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

Example:

Consider the language

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

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:

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

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

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

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

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

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

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

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

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

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

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 <u>complement class of C</u>.

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:

$UNSAT := \{ \varphi$ a formula in CNF $| \varphi$ is not satisfiable $\}$.

Then

$UNSAT \in$ co-NP, because

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

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 single 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 <u>closed under complement</u>, if for all $L \in C$ we have $\overline{L} \in C$.

Claim:

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

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

Lemma:

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

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

Proof:

- 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 space usage is bounded by the time.