4. Space vs. Time, Non-Determinism vs. Determinism

Goal: Show that $NTIME(t(n)) \subseteq DSPACE(t(n))$
and $NSPACE(s(n)) \subseteq DTIME(2^{O(s(n))})$.

4.1 Constructible Functions and Configuration Graphs

Goal:
- Define a notion of constructible functions that can be computed in a time/space economic way
- Define the state space (aka. configuration graph) of a Turing machine.

Definition:
Let $s, t : \mathbb{N} \to \mathbb{N}$ with $t(n) \geq n$.

- Function $t(n)$ is time constructible, if there is an $O(t)$-time-bounded DTM that computes the function $1^n \mapsto bin(t(n))$.
  Computing the function means that the result value is supposed to appear on a designated output tape when the machine enters its accepting state.
  The output tape is write-only.
- Function $s(n)$ is space constructible, if there is an $O(s)$-space-bounded DTM that computes the function $1^n \mapsto bin(s(n))$.

Examples: All functions of interest to us are time and space constructible, e.g. n, $n \log n$, n^2, 2^n.

Let M be a TM (deterministic or non-deterministic).

- The <u>configuration graph of M</u>

$$CG(M) := (Conf(M), \to_M)$$

consists of
- the <u>set of all configurations of M</u>, denoted by $Conf(M)$,
- together with the <u>transition relation among configurations</u> $\to_M$.

- This graph is typically infinite.
However, to find out whether M accepts an input $x \in \Sigma^*$,
all we have to do is find out
whether we can reach an accepting configuration
from the initial configuration $c_0(x)$.
The task is undecidable in general,
but it becomes feasible when the TM is time or space bounded.

<u>Lemma</u>:
Let M be an $s(n)$-space-bounded TM (non-deterministic or deterministic)
with $s(n) \geq \log n$ for all $n \in \mathbb{N}$.
There is a constant c (depending only on M, not on the input)
so that M on input $x \in \Sigma^*$ can reach
at most $c^{s(|x|)}$ configurations from $c_0(x)$.

<u>Proof</u>: Let $M = (Q, \Sigma, \Gamma, \Sigma, \sqcup, \delta, q_0, q_{accept}, q_{reject})$
be a k-tape DTM or NTM.
A <u>configuration of M</u> is described by
↳ the current state

↳ the content of the work tapes
↳ the position of the heads on the work tapes
↳ the position of the head on the input tape.

Thus, the number of configurations is at most

$$|Q| \cdot (|\Gamma|^{s(|x|)})^k \cdot s(|x|)^k \cdot (|x|+2).$$

This number in turn is bounded by

$$c^{s(|x|)}$$

for some constant c depending on $|Q|$, $|\Gamma|$, and k.
Note that we need $s(n) \geq \log n$ to bound $|x|+2$. □

Since we require deciders to always halt,
an immediate consequence of this estimation is the following.

Lemma:
Let $s(n) \geq \log n$ for all $n \in \mathbb{N}$.

$$\bigcup_{N} SPACE(s(n)) \subseteq \bigcup_{N} TIME(2^{O(s(n))}).$$

Proof:
Let $L \in NSPACE(s(n))$ with $L = L(M)$
for an NTM M that is a decider and $s(n)$-space bounded.
If M repeated a configuration, it would enter an infinite loop.
This contradicts the assumption that M terminates.
Hence, the running time of M is bounded by $c^{s(n)} \in 2^{O(s(n))}$
by the previous Lemma. □

Space-constructible functions are particularly interesting
because they can be used to enforce termination.

Intuitively, under the assumption of space constructibility the TM knows the space bound it is growing under.

Definition ($s(n)$-weak recognizer):
Let $s(n) \geq \log n$ be space constructible
and consider $L \subseteq \Sigma^*$.
We say that an NTM M is an *$s(n)$-weak recognizer of L*,
if
- $L = L(M)$ and
- for every $w \in L$ there is an accepting path

$$c_0(x) \to \dots \to c_m$$

with $\text{space}(c) \leq s(n)$ for all configurations c on the path.

Proposition (Self-Timing Technique):
Let $s(n) \geq \log n$ be space constructible
and consider $L \subseteq \Sigma^*$.
If there is an $s(n)$-weak recognizer M of L,
then there is an NTM M' that decides L in space $O(s(n))$.

Proof:
- By the assumption, every $w \in L$ has an accepting path

$$c_0(x) \to \dots \to c_m$$

with $\text{space}(c) \leq s(n)$ for all configurations c on the path.
While there are duplicate configurations on the path

$$c_0(x) \to \dots \to c_i \to \dots \to c_j \to \dots \to c_m,$$

($c_i = c_j$)

cut them out:

$$c_0(x) \to \dots \to c_i \to c_{j+1} \to \dots \to c_m.$$

The result is an accepting path of length $\leq c^{s(n)}$
by the previous lemma.
In short:
If we know that M has an accepting path
where the space usage is bounded,
then M also has a short path.

- We construct M' to simulate M for $c^{s(n)}$ steps.
If M accepts/rejects before this bound is reached,
M' will accept/reject.
If the bound is reached, M' will reject.
- Besides checking the length of the computation,
M' will make sure that the computation obeys the space bound.
To this end, it initially marks $s(n)$ cells on the work tape
(can be done because $s(n)$ is space constructible).
If these $s(n)$ cells are left, M' rejects.

↳ Together, M' will still accept L
as it will find all short and $s(n)$-space bounded
accepting computations.

- To enforce termination after the time bound,
the idea is to add to M a counter
that keeps track of how many steps
have been taken so far.
- The space used by M' on input $x \in \Sigma^*$ with $|x| = n$
is ↳ the space used by M on that input
↳ plus the additional space used to store the counter.

We assume that M uses space bounded by $s(n)$.
The counter requires

$$\log c^{s(n)} \in O(s(n)).$$

□

Remark:
A consequence of the self-timing technique is that we could have defined the complexity classes in a more liberal way, namely via weak recognizers.

4.2 Stronger Results

Goal:
- Improve $NSPACE(s(n)) \subseteq NTIME(2^{O(s(n))})$ to $NSPACE(s(n)) \subseteq DTIME(2^{O(s(n))})$.
- Improve $NTIME(t(n)) \subseteq NSPACE(t(n))$ to $NTIME(t(n)) \subseteq DSPACE(t(n))$.

Theorem:
$NTIME(t(n)) \subseteq DSPACE(t(n))$.

Proof:
- We do a depth-first search on the computation tree of the given $t(n)$-time-bounded NTM.
  To be precise, we construct the tree on-the-fly and accept if an accepting configuration is encountered.
- It may seem we need $O(t(n)^2)$ space to keep the stack of configurations for the depth-first search:
  ↳ each configuration requires $t(n)$ space
  ↳ and also the depth of the tree is $t(n)$.

· But we can do better:

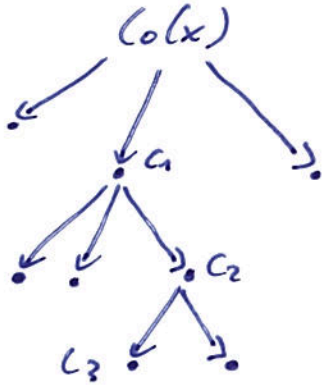
↳ we only have to store the path
from the initial configuration
to the configuration being currently visited.

↳ Assuming that all non-deterministic choices
are at most k-ary (k depends on M)
we can reconstruct the current configuration
in $t(n)$ space:

↳ start with the initial configuration $C_0(x)$,

↳ simulate the computation of the NTM M
using the k-ary string to resolve choices. □

Illustration:



Store 0 instead of C_3
2 C_2
1 C_1
0 $C_0(x)$.

Theorem:

Let $s(n) \geq \log n$.

$$NSPACE(s(n)) \subseteq DTIME(2^{O(s(n))}).$$

Proof.

↳ To begin with, we assume that $s(n)$ is space constructible.
We use this assumption to determine
the set of all configurations that use space at most $s(n)$.
To enumerate the configurations, we encode them as strings.

↳ The states are given numbers and we write them down in unary.

↳ To separate the components of a configuration we use a fresh symbol.

↳ The strings encoding each configuration have a length $\leq d \cdot s(n)$ for some constant d.

(More on this encoding in the next lecture.)

For the actual enumeration,

↳ we first mark $d \cdot s(n)$ cells (can be done by space constructibility)

↳ We use these cells as reference to enumerate all $d \cdot s(n)$ long strings in lexicographic order.

↳ For each of the enumerated strings, we check whether it is a configuration.

Altogether, the enumeration can be done in $2^{O(s(n))}$.

On the resulting set of configurations, we do a reachability check.
There are various options:

(1) Do a least fixed point and mark the reachable configurations.

(2) Also enumerate the edges and do a graph traversal.

↳ To get rid of the assumption of space constructibility, we do the procedure for a fixed space bound $s = 0, 1, 2, \ldots$
If we encounter a configuration that wants to use more space, we set $s := s+1$
We eventually hit $s(n)$ in which case no configuration will try to use more space.

- The time requirement is $\sum_{s=0}^{s(n)} d^s = \frac{d^{s(n)+1} - 1}{d-1} \in 2^{O(s(n))}$.

□