5. Alphabet Reduction, Tape Reduction, Compression, and Speed-Up

Goal:
- Show that the definition of the basic complexity classes is robust in the sense that it does not depend on the details of the Turing machine definition.
- These details are
  - ↳ the tape alphabet
  - ↳ the number of tapes
  - ↳ constant factors.

Consequence: We do not have to be too accurate about these details.

Technically: Show that a 1-tape TM over $\{\$, \sqcup, 0, 1\}$ can simulate the other features and do so efficiently.

The notion of simulation is the following.

Definition:
Consider TMs M and M' over input alphabet Σ.
Then M' is said to simulate M,
if $\forall x \in \Sigma^*: \quad x \in L(M) \iff x \in L(M')$.

Lemma (Alphabet Reduction):
Assume $M = (Q, T, \Sigma, \$, \sqcup, \delta, q_0, q_{accept}, q_{reject})$
is a decider that is $t(n)$-time bounded.
Then there is a decider $M' = (Q', \{\$, \sqcup, 0, 1\}, \{0,1\}, \$, \sqcup, \delta', q_0, q_{accept}, q_{reject})$
that is $4 \log |T| \cdot t(n)$-time bounded and satisfies

$\forall x \in \Sigma^*: \quad x \in L(M) \quad$ iff $\quad bin(x) \in L(M')$.

Here, $bin(-)$ is a fixed binary encoding for the letters in Γ.

Moreover, • M' will be deterministic iff M is deterministic.

• M' will use k types iff M uses k types.

Proof:

TM M' mimics the operation of M on the binary encoding of the symbols.

To be precise, it will

(1) use $\log |\Gamma|$ steps to read from each type the $\log |\Gamma|$ bits encoding a symbol of Γ.

(2) use its local state to store the symbols it has read

(3) use M's transition function / relation to compute the symbols that M writes and the state that M enters

(4) store this information in the control state of M'

(5) use $\log |\Gamma|$ steps to write the encoding into the types.

Concerning the size, one can check that the number of states of M' is

$$\leq C \cdot |Q| \cdot |\Gamma|^k \log |\Gamma|.$$

$|Q|$ = state of M

$|\Gamma|^k$ = one symbol per type, say k-types

$\log |\Gamma|$ = counter from 1 to $\log |\Gamma|$.

□

For the type reduction, we first show a general construction that we then analyse wrt. its time and space usage.

Theorem (Tape Reduction):

For every k-tape TM M there is a 1-tape TM M' that simulates M.
Moreover, M' is deterministic if M is deterministic.
If M uses an additional input type, M will use an additional input type.

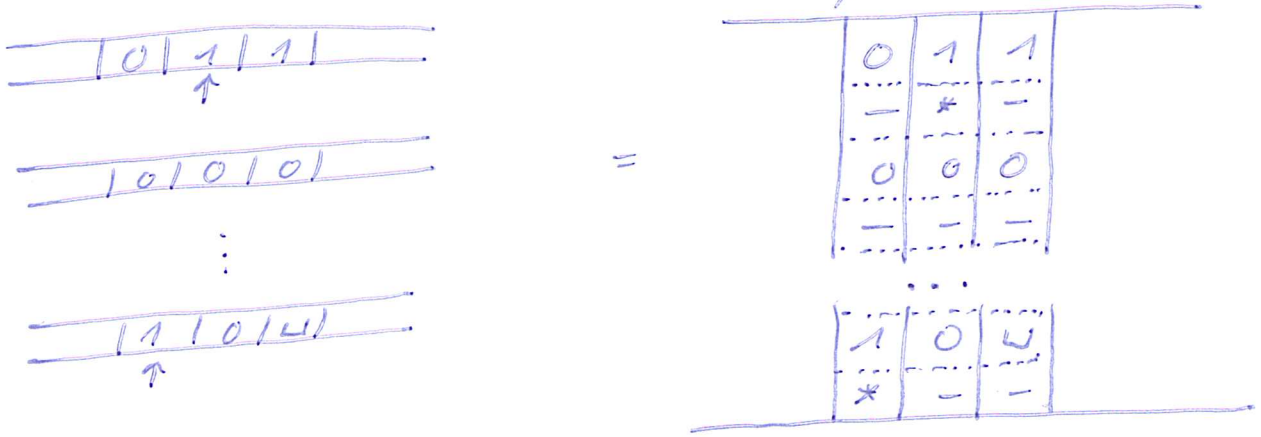
Proof:

M' simulates one step of M by a sequence of steps.
The idea is to store the k-types into 1-type.
This one type will be understood as divided into $2k$-tracks.
To be precise, the type alphabet is

$$T' := (T \times \{*, -\})^k \cup \Sigma \cup \{\#, \sqcup\}.$$

The $(2\ell-1)$st component of a letter in T' store the content of the ℓth type.
The (2ℓ)th component (in $\{*, -\}$) marks the position of the head on the ℓth type by $*$.
There will be precisely one $*$ on the (2ℓ)th track, the remaining symbols are $-$.

To illustrate the construction:



k steps of M is simulated by M' as follows:

- M' always starts on the $\#$ (left endmarker)
- It moves to the right until it finds the first $\sqcup$
  (note that $\sqcup$ is neither a vector containing $\sqcup$ nor a vector with $-$, $\begin{pmatrix}\vdots\\ \sqcup\\ \vdots\end{pmatrix}$ nor $\begin{pmatrix}\vdots\\ -\\ \vdots\end{pmatrix}$).
- On the way, M' stores the k symbols.
- Once M' has collected all k symbols, it can simulate a transition of M.
- It moves back to the start and, on its way, makes the changes to the type that M would make.
- When arriving at $\#$, M' changes the control state. □

Definition:

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

Define

$$\mathrm{DTIMESPACE}_k := \{ L(M) \mid M \text{ is a } k\text{-type } \mathrm{DTM}_N \text{ (with extra input type)} \text{ that is a decider and that is } t(n)\text{-time and } s(n)\text{-space bounded} \}$$

We drop the index if there is only one work type.

Lemma:

For all $t, s: \mathbb{N} \to \mathbb{N}$:

$$\mathrm{DTIMESPACE}_k(t, s) \subseteq \mathrm{DTIMESPACE}(O(ts), s)$$

Proof:

Consider $L(M) \in DTIMESPACE_k(t,s)$.

- The TM M' from the above theorem simulates each step of M by $O(s(n))$ steps.
  Since M makes at most $t(n)$ steps, M' makes at most $O(t(n) \cdot s(n))$ steps.
- M' does not use more space than M does. □

Corollary:

For all $t: \mathbb{N} \to \mathbb{N}$, we have:

$$D_N TIME_k(t) \subseteq \bigcap_{\varepsilon > 0} D_N TIME(O(t^{1+\varepsilon})).$$

Proof:

Let N be a $t(n)$-time bounded DTM.
We observed in the last lemma from Section 2, that in $t(n)$-steps a TM can visit only $t(n)$-cells.
Thus $Space_N(n) \leq t(n)$, and the claim follows from the above lemma. □

Remark (Oblivious Turing Machines):

The construction in the above theorem can be modified to ensure that M' is oblivious:
Its head movement does not depend on the input, but only depends on the input length.
More formally:
For every $x \in \Sigma^*$ and $i \in \mathbb{N}$, the location of each of M's heads at the ith step of execution on input x is only a function in $|x|$ and i.

The fact that every TM can be simulated by an oblivious TM will simplify some proofs.
Requires constructible functions (next week).

The tape compression result shows that
we do not need to care about constant factors:

Every TM M can be simulated by a TM M'
that uses only a constant fraction of the space used by M.

Lemma (Tape compression):
For all $0 < \varepsilon \leq 1$ and all $s: \mathbb{N} \to \mathbb{N}$:

$$\mathrm{DSPACE}(s(n)) \subseteq \mathrm{DSPACE}(\lceil \varepsilon \cdot s(n) \rceil).$$

Idea: The statement can be understood as being the converse
of the alphabet reduction lemma.
Rather than distributing one symbol to several cells,
we enlarge the tape alphabet to store several symbols in one symbol.
All that remains is to simulate.
(The idea can be compared to having a 64 bit architecture
that is able to store more information per cell
than an 8 bit architecture.)

Proof: Let $c := \lceil \frac{1}{\varepsilon} \rceil$ and let M be a 1-tape DTM.
We simulate M by a DTM M' with tape alphabet

$$\Gamma' := \Gamma^c \cup \Sigma \cup \{\emptyset, \sqcup\}.$$

A block of c cells is encoded into one cell of M'.
So instead of s cells, M' only uses

$\lceil \frac{s}{c} \rceil \leq \lceil \varepsilon \cdot s \rceil$ cells. (For the inequality, note that

$$c \geq \frac{1}{\varepsilon} \geq 1 \Rightarrow \frac{s}{c} \leq \varepsilon \cdot s$$

$$\Rightarrow \lceil \frac{s}{c} \rceil \leq \lceil \varepsilon \cdot s \rceil.$$

M' can simulate M step by step.
To this end, M' stores the position of the head
inside a block of cells in its control state.
If M moves its head inside a block,
M' does not move the head but only changes
the symbol and the control state.

If M moves its head from a cell c in one block
to cell $c+1$ in another block, also M' moves its head. □

We now show a similar compression trick for time.

Claim (Linear Speed-Up, Hennie & Stearns '65):
For all $k \geq 2$ and all $\varepsilon: \mathbb{N} \to \mathbb{N}$
and all $0 < \varepsilon \leq 1$:

$$\bigcup_N \mathrm{TIME}_k(t(n)) \subseteq \bigcup_N \mathrm{TIME}_k(n + \varepsilon(n \cdot t(n))).$$

Idea: As before, the idea is to store $c \in \mathbb{N}$ cells into one new cell.
Formally, we will copy the input and compress it.
This costs $n + \varepsilon n$ steps (read from left to right,
go back to beginning).

To get the speed-up, M' now has to simulate
c-steps of M with just a single step.

↳ If M stays within the c-cells of one block,
this is no problem:
we can just precompute the outcome of the c-steps.

↳ Problematic is the case when M moves back and forth
between two cells that belong to neighboring blocks.

Solution: M' stores three blocks in its finite control,
the current block B, the block to the left of B,
and the block to the right of B.

With these three blocks, M' can simulate any c-steps of M
in its finite control.

Only afterwards M' updates the tape content.
If M left block B, then M' has to update
the blocks in its finite control. □