10. Models of Computation for L and NL

Goal: • Show that the following classes of automata can simulate each other (both in the deterministic and in the non-deterministic case):

- ↳ Logspace-bounded TMs
- ↳ k-Counter two-way automata with linearly bounded counters
- ↳ k-Head two-way finite automata
- ↳ Logspace-bounded DTMs with polynomial read-once certificates.

• This shows that the corresponding ressources are equally powerful (but sometimes, counters may be more convenient than tape)

• The relationship carries over to higher space complexity classes.

Definition:

• A k-counter two-way automaton (kCA) is a tuple $A = (\Sigma, Q, C, \rightarrow, q_0, q_f)$

with

$$\rightarrow \subseteq Q \times \Sigma \times \{L, R\} \times \underbrace{\mathbb{P}(C)}_{\text{to be tested for being zero}} \times \underbrace{\mathbb{P}(C)}_{\text{add 1}} \times \underbrace{\mathbb{P}(C)}_{\text{subtract 1}} \times Q$$

• The semantics of a kCA A with input x is defined in terms of configurations from

$$Conf_x^A := Q \times \underbrace{\mathbb{Z}^C}_{\text{counter values}} \times \underbrace{[1, |x|]}_{\text{Head position}}$$

• Given input x, the transition relation $\rightarrow \subseteq Conf_x^A \times Conf_x^A$ among configurations is defined as expected (the input is read only).

• In the linearly-bounded semantics, given input x, if a counter has value $|x|$ (or $-|x|$) transitions that increment (or decrement) this counter are disabled.

Theorem (Minsky '67): 2CA are Turing complete.

With the linearly-bounded semantics, we arrive at L / NL.

Definition:

A k-head two-way finite automaton is a finite automaton with k-heads into the input.

There is no work tape.

The input is read only.

Theorem:

A language L

(1) is decided by a logspace-bounded DINTM

iff (2) it is decided by a DIN k-counter two-way automaton with linearly bounded counters

iff (3) it is decided by a k-head two way DINFA.

Proof. We show (1) => (2) and (3) => (1), (2) => (3) is immediate.

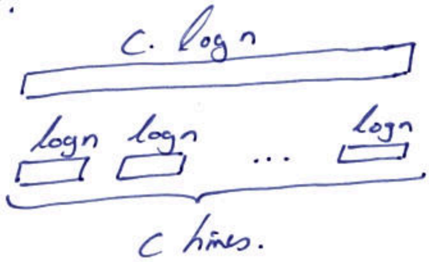
(1) => (2):

- Let N be an O(log n)-space-bounded 1-(work-) tape DINTM. Assume the work tape alphabet is {0,1}. Assume the values are non-negative integers. We simulate N by a CA A where the values are non-negative integers.
- In a first step, we show how to implement the following operations:
  - ↳ Duplicate the value of a counter c: Zero-out two scratch counters d and e. Repeatedly decrement c while incrementing d and e.
  - ↳ Double the value of a counter c: Repeatedly decrement c while incrementing d twice.

↳ Halve the value of a counter c:
Similar to double.
↳ Check whether the value of a counter c is even:
Duplicate c and repeatedly subtract 2 from one copy.
See whether the process leaves 1 rest.
↳ Add or subtract the value of one counter from another counter:
To add, increment one counter while decrementing the other.
To subtract, decrement both.

Mimicking Configurations:

· Given a configuration of N, the work tape content
can be understood
as a $c \cdot \log n$ - bit binary number.
We break this number up into
c blocks of log n bits each.



The log n - bit numbers are stored
in c counters of A.

· Another counter of A is used to store
the position of N's work tape head:
↳ The block scanned by N
is stored in A's finite control.
↳ The position i within the block
is represented by 2^i in the counter.

· The position of the head into the input coincides for N and A.
The control state also coincides for N and A.

· There is a finite number of scratch counters.

Simulating N:

- To simulate a move, A must know the symbols being scanned by N on the two tapes.
  - ↳ It can read the input head directly.
  - ↳ For the work tape, A must determine and change the ith-bit of a number stored in a counter c. Note that 2^i is stored in another counter d.
    - First duplicate c and d so not to lose their contents.
    - Repeatedly halve c and d until d contains 1. Now check whether c is even. This is (even = 0, odd = 1) the ith bit of the original c.
    - We can modify the bit by adding or subtracting the original value of d from the original c.

(3) ⇒ (1):

- Given a k-head two-way DINFA, we construct an $O(\log n)$-space-bounded DINTM N that simulates A.

Mimicking Configurations:

- The work tape of N is partitioned into k-tracks, each holding a binary number in $[-(n+1), n+1]$. The numbers represent the positions of the k simulated heads of A relative to the position of N's read head (initially zero, the read head is all the way to the left)
- The state of A is the state of N.

Simulating A:

- To simulate a move of A, N needs to know the symbols under each head of A.

- Starting from its read head all the way left,
  N moves its head to the right
  and decrements each of the counters.
- Whenever a counter contains 0,
  the head of M corresponding to that counter
  is scanning the input tape cell that N is currently scanning.
  N reads the symbol and remembers it in its finite control.
- When N has reached the right side of the input tape,
  it has seen all symbols under the k simulated heads of M.
  It changes the counters and the control state
  according to the transition relation of M
  (held as part of N's finite control).
- Now N moves back to the left,
  incrementing the counters as it goes,
  and simulates the next step of M. □

<u>Idea of certificates:</u>

- NL and NP have alternative definitions
  that <u>replace non-determinism</u> with the notion of
  a <u>certificate for membership</u>.

<u>Example</u>: A certificate for PATH is the sequence of nodes.

<u>Intuitively</u>: The certificate resolves non-deterministic choices of an NTM.

<u>Problem</u>:
- In the case of NL, the certificate may be polynomially long.
- So a logspace machine may not have the space to store it.

<u>Solution</u>: The certificate is provided on a <u>read-once</u> input tape
that is not counted towards the machine's space usage.

Write and read once and only:

- When we defined the output tape of a TM, we called it write-only:
  - ↳ If the TM writes something, it moves its head to the right.
  - ↳ The TM may decide not to write in a step.

  We could have called this model write-once, and both terms are used in the literature.
- Alternatively, and equally powerful, we could have forbidden the TM to ever move left.
- By read-once we could also mean
  - ↳ read and move right

  or
  - ↳ never move left.

  We stick with the former restriction.

Theorem:

A language A is in NL iff there is a logspace-bounded DTM M with read-once input tape, called the verifier, and a polynomial $p: \mathbb{N} \to \mathbb{N}$ so that for all $x \in \Sigma^*$

$$x \in A \quad \text{iff} \quad \exists \underbrace{u \in \{0,1\}^{p(|x|)}}_{\text{the certificate}} : M(x,u) = 1.$$

Here, $M(x,u)$ is the output of M when started with
- x on its input tape and
- u on its read-once tape.

Proof (Sketch):

"only if"
- Let $A \in NL$ be decided by the logspace-bounded NTM N. Wlog. choices can be assumed to be binary.
- Whenever N makes a choice, we add a bit to the certificate.

Since N runs in polynomial time,
we obtain a polynomially long string.

- The verifier M is based on N but modified as follows.
  Whenever N makes a non-deterministic choice,
  M looks up the certificate.

"if" Given a verifier M for A,
we turn it into an NTM N for A
that guesses the certificate bit.

$\square$

<u>Note:</u> If we remove the read-once restriction
(can read certificate-bits several times),
we arrive at a characterization of NP.

<u>Question:</u> What is a certificate
- for 2SAT
- for 3SAT

?