Hierarchy theorems

Anton Bankevich

Part 1

In this part we shall prove
the hierarchy theorems for
the following classes of languages:

- DTime(f(n))
- DSpace(f(n))
- NTime(f(n))

Constructible functions

- Def. f: N → N is time constructible function iff there is a DTM which if given an input consisting of 1^n constructs f(n) and prints it on the output tape in f(n) time.
- Def. f: N → N is space constructible function iff there is a DTM which if given an input consisting of 1^n constructs f(n) and prints it on the output tape using f(n) space.

DTime

- Def. DTime(f(n)) is a set of languages which can be decided by a DTM in f(n) steps. Here f(n) is a time constructible function.

DTime hierarchy theorem

Th1. If

- $f(n)$, $g(n)$ are time constructible functions
- $f(n)\log(g(n)) = o(g(n))$
- $f(n) > n$ for sufficiently large n.

Then $DTime(O(f(n))) \neq DTime(O(g(n)))$.

Proof of Theorem 1

- We shall construct a DTM (we call it D), such that
- for any Turing Machine M from DTime(O(f(n)) one can find an input string S which is accepted by D iff M rejects S.

Proof of Theorem 1: Enumeration

- Every Turing Machine is just a number of rules which state what we should do in a given configuration.
- These rules can be written as a string in

alphabet {“0”, “1”}.

These statements give the needed enumeration.

Let now M_k be the Turing Machine which corresponds to k in our enumeration (k is a number whose binary representation is our code of DTM).

Proof of Theorem 1: Construction

Let us give S to D as an input

- If $S \neq k\$1^l$ for certain k and l, then D accepts S (in $k\$1^l$, k is the binary representation of k).
- If $S = k\$1^l$, then D simulates M_k on S for h(n) steps. h(n) will be defined later.
  - If M_k doesn't exist, D accepts S.
  - If M_k halts in this time and accepts input, D rejects S.
  - If M_k halts in this time and rejects input, D accepts S.
  - If M_k doesn't halt in this time, D accepts S.

Diagonalization

- The construction which we have used above is called diagonalization: one machine constructs and simulates all machines from a certain class.

Proof of Theorem 1: D belongs to DTime(g(n))

Now we are to count the time of evaluation of D.

- Checking of $S = k\$1^l$ takes O(n) time.
- Simulation of M_k takes O(h(n)log(h(n)) time.

So D operates during O(n) + O(h(n)log(h(n)) time => h must be such that $h(n)\log(h(n)) \leq g(n)$.

Proof of Theorem 1: D doesn't belong to DTime(O(f(n)))

- Let M_k belong to DTime(f(n)). Then M_k operates for c·f(n) time, at most.
- Let $S = k\$1^l$.
- If f(n) = o(h(n)), then for sufficiently large n h(n) > c·f(n). It means that simulation of M_k halts in this time and D yields an answer opposite to M_k on $k\$1^l$. So M_k isn't equal to D for any M_k from DTime(f(n)).

Proof of Theorem 1: h(n)

To complete the proof we are to find a function h(n) :

$$h(n)\log(h(n)) \le g(n)$$

$$f(n) = o(h(n))$$

$$\text{Let } h(n) = \frac{g(n)}{2\log(g(n))}$$

Theorem 1'.

Th1'. If

- $f(n)$, $g(n)$ are time constructible functions
- $f(n)\log(f(n)) = o(g(n))$
- $f(n) > n$ for sufficiently large n.

Then $DTime(f(n)) \neq DTime(g(n))$.

Proof of Theorem 1'

We have proved that

- D belongs to DTime(g(n)) and
- D doesn’t belong to DTime(O(f(n))).

This is enough to prove our theorem.

DSpace

- Def. DSpace(f(n)) is a set of languages which can be decided by DTM using f(n) space. Here f(n) is a space constructible function.
- Def. $f: N \to N$ is space constructible function iff there is a DTM which if given a number n, constructs f(n) and prints it on the output tape using f(n) space.

DSpace hierarchy theorem.

Th2. If

- $f(n)$, $g(n)$ are space constructible functions
- $f(n) = o(g(n))$
- $f(n) > \log(n)$ for sufficiently large n.

Then $DSpace(f(n)) \neq DSpace(g(n))$.

Proof of Theorem 2: Construction

Let us give S to D as an input

- If $S \neq k\$1^l$ for certain k and l, then D accepts input.
- If $S = k\$1^l$, then D simulates M_k on S for $h(n) \cdot 2^{h(n)}$ steps and while not more than h(n) space is used.
  - If M_k doesn't exist, D accepts S.
  - If M_k halts after using not more than a given space and time and it accepts input, D rejects S.
  - If M_k halts after using not more than a given space and time and it rejects input, D accepts S.
  - If M_k doesn't halt in time or it uses extra space , D accepts S.

Proof of Theorem 2: D belongs to DSpace(g(n))

- In the first part:

  If $S \neq k\$1^l$ for certain k and l, then D accepts input.

D uses O(1) space.

- In the second part:

  If $S = k\$1^l$, then D simulates M_k on S for $h(n) \cdot 2^{h(n)}$ steps and while not more than h(n) space is used.

D uses log(n) + h(n) space.

Proof of Theorem 2: D doesn't belong to DSpace(O(f(n)))

- D is Turing Machine, consequently for some k D is equal to M_k.
- Let M_k belong to DSpace(O(f(n))). Then D operates with c·f(n) space, at most.
- Let S = $k\$1^l$.
- If f(n) = o(h(n)), then for sufficiently large b h(n) > c·f(n). It means that simulation of M_k halts using only the given space and D yields an answer opposite to M_k on $k\$1^l$. So M_k isn't equal to D for any M_k from DTime(f(n)).

Proof of Theorem 2: $h(n)$

Conditions for $h(n)$:

- $\log(n) + h(n) < g(n)$
- $f(n) = o(h(n))$

Let $h = g(n) - 2\cdot\log(n)$. This satisfies all conditions.

Theorem 2'

Th2 '. If

- $f(n)$, $g(n)$ are time constructible functions
- $f(n)\log(f(n)) = o(g(n))$
- $f(n) > n$ for sufficiently large n.

Then $DSpace(O(f(n))) \neq DSpace(O(g(n)))$.

NTime(f(n))

- Def. NTime(f(n)) is a set of languages which can be decided by NTM in f(n) steps. Here f(n) is a time constructible function.

NTime hierarchy theorem

Th3. If

- $f(n)$, $g(n)$ are time constructible functions
- $f(n) = o(g(n))$
- $f(n) > n$ for sufficiently large n.

Then $NTime(O(f(n))) \neq NTime(O(g(n)))$.

NTime hierarchy theorem: Remark

- The technique from two previous theorems cannot be directly applied here. We can’t return the opposite answer as M_i, because M_i is NTM.

Proof of Theorem 3: Enumeration

- As in Theorems 1 and 2, we can enumerate all NTMs in such a way that we can evaluate this machine in the run time.

Proof of Theorem 3: Assumption

I shall prove the theorem for the case

- $f(n) = n$
- $g(n) = n^2$

Proof of Theorem 3: Function N(i)

$$\text{Let } N(i) = 2^{2^{2^{2^{2^{i}}}}}$$

For any given k we can find $N^{-1}(k)$: such integer i that $N(i) < k \le N(i+1)$ in O(k) time.

Proof of Theorem 3: Construction

Let us give S to D as an input

- If $S \neq 1^n$ for any n, then D accepts input.
- If $S = 1^n$ then
  - D computes $i = N^{-1}(n)$
  - If M_i doesn't exist, D accepts S.
  - If $n \neq N(i+1)$, D simulates M_i on the string 1^{n+1} with the help of nondeterminism for $n^{1.5}$ steps.
    - If M_i halts in time, D outputs the answer of M_i.
    - If M_i doesn't halt in time, D accepts S.
  - If $n = N(i+1)$, D simulates M_i on the string $1^{N(i)+1}$ for $n^{1.5}$ steps, checking all brunches.
    - If M_i halts in time, D outputs an answer opposite to M_i .
    - If M_i doesn't halt in time, D accepts S.

Proof of Theorem 3: D belongs to NTime($O(n^2)$)

- If $S \neq 1^n$ for any n, then D accepts input.

This part of the algorithm operates in O(n) steps.

  - D computes $i = N^{-1}(n)$

This part of the algorithm operates in O(n) steps.

  - If $n \neq N(i+1)$, D simulates M_i on the string 1^{n+1} with the help of nondeterminism for $n^{1.5}$ steps.
  - If $n = N(i+1)$, D simulates M_i on the string $1^{N(i)+1}$ for $n^{1.5}$ steps, checking all brunches.

This part of the algorithm operates in $O(n^{1.5})$ steps.

Proof of Theorem 3: D doesn't belong to NTime(O(n))

- Let D be equal to M_i.
- Suppose D belongs to NTime(O(n)).
- Then D operates during c·n time, at most.

Proof of Theorem 3: D doesn't belong to NTime(O(n))

- Let S = 1^k. Where k runs through all values $N(i) < k \le N(i+1)$.
- For all $N(i)<k<N(i+1)$ M_i halts in time and on 1^{k+1} should give the same answer as D on 1^k.
- For $k=N(i+1)$ M_i works in time $O(2^{(N(i)+1)^2})$, which is less than $N(i+1)$.So D gives an answer opposite to M_i on $1^{N(i)+1}$.

Proof of Theorem 3: D doesn't belong to NTime(O(n))

But $M_i = D$, so

$$M_i(\ 1^{N(i)+1}\ ) = D(\ 1^{N(i)+1}\ ) =$$
$$= M_i(\ 1^{N(i)+2}\ ) = D(\ 1^{N(i)+2}\ ) =$$
$$= M_i(\ 1^{N(i)+3}\ ) = \ldots = D(\ 1^{N(i+1)-1}\ ) =$$
$$= M_i(\ 1^{N(i+1)}\ ) = D(\ 1^{N(i+1)}\ ),$$

which contradicts definition of $D(1^{N(i+1)})$

Part 2

- In this part we shall discuss conditions of constructability of functions in the hierarchy theorems.

Gap Theorem

Th4. There exists such a function f: $N \rightarrow N$ that:

- f can be calculated using a certain DTM (without time or space bounds).
- $DTime(f(n)) = DTime(2^{f(n)})$
- $f(n) > n$ for enough large n.

Proof of Theorem 4: Enumeration

- Let M_0, M_1, M_2, …be an enumeration of all Turing Machines.

Proof of Theorem 4: P(i, k) property

Let P(i, k) be the following property:

Each machine among M_0, M_1,… M_i on each input of length i will either halt after fewer than k steps or it will halt after more than 2^k steps or it will not stop at all.

Proof of Theorem 4: Sequence k_i

Let $k_j(i)$ be a sequence such that:

$$k_1(i) = 2i$$

$$k_{j+1}(i) = 2^{k_j} + 1$$

Let f(i) = min($k_l(i)$: P(i, $k_l(i)$))

Proof of Theorem 4: Decision of f(i)

- There is a finite number of inputs of length i.

$$N(i) = \sum_{k=0}^{i} |\Sigma_k|^i$$

- Here Σ_k is the alphabet of M_k.
- So f(i) can be decided by the following algorithm:
- Find $P(i, k_l(i))$ for all $1 < l < N(i)+2$.

f(i) will be the minimal $k_l(i)$such that $P(i, k_l(i))$ holds.

Proof of Theorem 4: Conclusion

- Let L be a language from DTime($2^{f(n)}$). Let M_j decide L.
- Only finitely many inputs (less then $j \cdot |\Sigma|^j$, where Σ is an alphabet of M_j), can stop after f(n) steps. These inputs can be checked in o(f(n)) time.
- So L belongs to DTime(f(n)).

Space theorems

- Th5 DSpace(O(log(log(n)))) $\neq$ DSpace(O(1)).

- Th6 For all $\varepsilon > 0$

  DSpace($O(\log(\log(n))^{1-\varepsilon}$)) = DSpace(O(1)).

Proof of Theorem 5: Lemma

Lm. Let A be a language from DSpace(O(1)). Then one can find n such that :

$\forall x \in A : |x| > n,$

$\exists a, b, c : x = abc$ and

$\forall r \in N, \ ab^r c \in A.$

Proof of Theorem 5: Construction

$$L = \{\underbrace{0...00}_{k}\$\underbrace{0...01}_{k}\$\underbrace{0...10}_{k}\$......\$\underbrace{1...11}_{k} \mid k \in N\}$$

L belongs to DSpace(O(log(log(n)))), but not to DSpace(O(1)).

Theorem 6

- Th6 For all $\varepsilon > 0$
  $DSpace(O(\log(\log(n))^{1-\varepsilon})) = DSpace(O(1))$.

Proof of Theorem 6: Pseudo Configurations

- Let a pseudo configuration be a configuration where, instead of positions of heads, only the symbols at these positions are taken into account.

Proof of Theorem 6: Sequences

- Let M be a DTM from

  DSpace($O(\log(\log(n))^{1-\varepsilon}$)).
- We will observe a sequence of pseudo configurations which occur with preset positions of heads.
- The number of different sequences is $o(n)$.

Proof of Theorem 6

One can find $N : \forall n > N$ the number of sequences for the input of length n is not greater than n/2.

We will show that on all inputs M uses less than $\log(\log(N))$ space.

Proof of Theorem 6

- Let x be the smallest input: M uses more than log(log(N)) space.
- X can be represented as $\alpha a \beta a \gamma a \delta$, where the marked symbols a have the same sequences of pseudo configurations.
- Then it is rather easy to show that if each pseudo configuration takes place during the evaluation of $M(\alpha a \beta a \gamma a \delta)$, then it must appear during the evaluation of $M(\alpha a \beta a \delta)$ or $M(\alpha a \gamma a \delta)$. This completes the proof.

Results

- We have proved three hierarchy theorems: for DTime, DSpace and NTime.
  - In DSpace and DTime $f = o(g)$ is sufficient.
  - In NTime $f \cdot \log(g) = o(g)$ is needed.

Results

- We have observed two cases of noncomputatible functions:
  - In the first case, f was very large and DTime(f) = DTime(2^f).
  - In the second case, f was very small and DTime(f) = DTime(1).

Thank you.

Any questions?