Nondeterministic Complexity Classes

1 Defining complexitiy classes

1.1 Syntax

def: nondeterministic 1-tape Turing machines (NTM)

$$M = (Z, \Sigma, \delta, z_0, Z_A)$$

- Z: finite set of states
- Σ: finite tape alphabet with

$$\{0, 1, \#, B\} \subseteq \Sigma$$

- set valued transition function

$$\delta : Z \times A \to 2^{Z \times \Sigma \times \{L,N,R\}}$$

where

$$(z', B, m) \in \delta(z, a)$$

means: if M reads a in state z it can print b, go to state z' and make the head move m.

- z_0 start state
- $Z_A \subseteq Z$: set of accepting end states

1 Defining complexitiy classes

1.1 Syntax

def: nondeterministic 1-tape Turing machines (NTM)

$$M = (Z, \Sigma, \delta, z_0, Z_A)$$

- Z: finite set of states
- Σ: finite tape alphabet with

$$\{0, 1, \#, B\} \subseteq \Sigma$$

- set valued transition function

$$\delta : Z \times A \to 2^{Z \times \Sigma \times \{L,N,R\}}$$

where

$$(z', B, m) \in \delta(z, a)$$

means: if M reads a in state z it can print b, go to state z' and make the head move m.

- z_0 start state
- $Z_A \subseteq Z$: set of accepting end states

1.2 Semantics

set K of configurations: as for deterministic machines.

successor configuration For $k, k' \in K$ configuration we say k' is *a* successor configuration of k and write $k \vdash k'$ if

- $$k = ubzav \ , a, b \neq \varepsilon$$

$$k' = \begin{cases} uz'bcv & (z', c, L) \in \delta(z, a) \\ ubz'cv & (z', c, N) \in \delta(z, a) \\ ubcz'v & (z', c, R) \in \delta(z, a) \end{cases}$$

- and so on. Always replace in deterministic case condition

$$\delta(z, a) = (z', c, m) \quad \text{by} \quad (z', c, m) \in \delta(z, a)$$

k-tape machines: :analogous.

computations: sequences of configurations (k_i) with

$$k_i \vdash k_{i+1} \quad \text{for all } i$$

computation accepting: if state of last configuration in $\mathbb{Z}_A$

M accepts w: if there *exists* and accepting computation of M started with w.

example: guessing a word $w \in \mathbb{B}^*$ and writing it on empty tape

$$\begin{aligned} Z &= \{z_0, z_1\} \\ Z_A &= \{z_1\} \\ \delta(z_0, B) = &= \{(z_0, 0, R), (z_0, 1, R), (z_1, B, N)\} \end{aligned}$$

example: guessing a word $w \in \mathbb{B}^*$ and writing it on empty tape

$$\begin{aligned} Z &= \{z_0, z_1\} \\ Z_A &= \{z_1\} \\ \delta(z_0, B) = &= \{(z_0, 0, R), (z_0, 1, R), (z_1, B, N)\} \end{aligned}$$

1.3 Resource Bounds

time and space used by a computation: as in deterministic case.

M **is** $t(n)$ **time bounded:** if for all $w \in L(M)$ *there exists an accepting computation* of machine *M* started with an input *w* which uses at most time $t(|w|)$.

M **is** $t(n)$ **space bounded:** if for all $w \in L(M)$ *there exists an accepting computation* of machine *M* started with an input *w* which uses at most space $t(|w|)$.

example: guessing a word $w \in \mathbb{B}^*$ and writing it on empty tape

$$\begin{aligned} Z &= \{z_0, z_1\} \\ Z_A &= \{z_1\} \\ \delta(z_0, B) = &= \{(z_0, 0, R), (z_0, 1, R), (z_1, B, N)\} \end{aligned}$$

1.3 Resource Bounds

time and space used by a computation: as in deterministic case.

M **is** $t(n)$ **time bounded:** if for all $w \in L(M)$ *there exists an accepting computation* of machine M started with an input w which uses at most time $t(|w|)$.

M **is** $t(n)$ **space bounded:** if for all $w \in L(M)$ *there exists an accepting computation* of machine M started with an input w which uses at most space $t(|w|)$.

example:

$$L = \{bin(x) \ : \ x \in \mathbb{N} \text{ is not a prime number}\}$$

non deterministic acceptor for M for L

- $bin(x)$ of length n
- guess factors $bin(x_1), bin(x_2)$ with $1 < x_i < x$. Time $O(n)$
- multiply and compare $x_1 \cdot x_2 = x$. Accept if this holds. With 2 tapes in time n^2

Machine M is $O(n^2)$-time bounded.

2 Complexity Classes

2.1 nondeterministic complexity classes

$NTIME_k(t(n)) = \{L : L \text{ accepted by an } O(t(n)) - \text{time bounded } k - \text{tape NTM}\}$

$$NTIME(t(n)) = \bigcup_k NTIME(t(n))$$

$NSPACE_k(t(n)) = \{L : L \text{ accepted by an } O(t(n)) - \text{space bounded } k - \text{tape NTM}\}$

$$NSPACE(t(n)) = \bigcup_k SPACE(t(n))$$

2 Complexity Classes

2.1 nondeterministic complexity classes

$NTIME_k(t(n)) = \{L : L \text{ accepted by an } O(t(n)) - \text{time bounded } k - \text{tape NTM}\}$

$$NTIME(t(n)) = \bigcup_k NTIME(t(n))$$

$NSPACE_k(t(n)) = \{L : L \text{ accepted by an } O(t(n)) - \text{space bounded } k - \text{tape NTM}\}$

$$NSPACE(t(n)) = \bigcup_k SPACE(t(n))$$

2.2 tape reduction

Lemma 1.

$$\begin{aligned} NTIME(t(n)) &\subseteq NTIME_1(t^2(n)) \\ NSPACE(t(n)) &\subseteq NSPACE_1(t(n)) \end{aligned}$$

Proof. similar to deterministic case

2 Complexity Classes

2.1 nondeterministic complexity classes

$$NTIME_k(t(n)) = \{L : L \text{ accepted by an } O(t(n)) - \text{time bounded } k - \text{tape NTM}\}$$

$$NTIME(t(n)) = \bigcup_k NTIME(t(n))$$

$$NSPACE_k(t(n)) = \{L : L \text{ accepted by an } O(t(n)) - \text{space bounded } k - \text{tape NTM}\}$$

$$NSPACE(t(n)) = \bigcup_k SPACE(t(n))$$

2.2 tape reduction

Lemma 1.

$$\begin{aligned} NTIME(t(n)) &\subseteq NTIME_1(t^2(n)) \\ NSPACE(t(n)) &\subseteq NSPACE_1(t(n)) \end{aligned}$$

Proof. similar to deterministic case

Exercise: show

$NTIME(t(n) \subseteq NTIME_2(t(n))$

Hint:

- guess the transitions of an accepting computation
- then verify for each tape separately, that in each step your guess of the symbol read on that tape is consistent with what was written in previous steps

2 Complexity Classes

2.1 nondeterministic complexity classes

$$NTIME_k(t(n)) = \{L : L \text{ accepted by an } O(t(n)) - \text{time bounded } k - \text{tape NTM}\}$$

$$NTIME(t(n)) = \bigcup_k NTIME(t(n))$$

$$NSPACE_k(t(n)) = \{L : L \text{ accepted by an } O(t(n)) - \text{space bounded } k - \text{tape NTM}\}$$

$$NSPACE(t(n)) = \bigcup_k SPACE(t(n))$$

2.2 tape reduction

Lemma 1.

$$\begin{aligned} NTIME(t(n)) &\subseteq NTIME_1(t^2(n)) \\ NSPACE(t(n)) &\subseteq NSPACE_1(t(n)) \end{aligned}$$

Proof. similar to deterministic case

Exercise: show

$$NTIME(t(n) \subseteq NTIME_2(t(n))$$

Hint:

- guess the transitions of an accepting computation
- then verify for each tape separately, that in each step your guess of the symbol read on that tape is consistent with what was written in previous steps

2.3 Some simple relationships

$$\begin{aligned} TIME(t(n)) &\subseteq SPACE(t(n)) \\ DTIME(t(n)) &\subseteq NTIME(t(n)) \\ DSPACE(t(n)) &\subseteq NSPACE(t(n)) \\ &\subsetneq ? \end{aligned}$$

Intuitively clear but ...

3 The cost of eliminating nondeterminism

3.1 time

the trivial simulation with exponential overhead

Lemma 2. *Let $t(n)$ be time constructible. Then*

$$NTIME(t(n)) \subseteq \bigcup_c TIME(2^{c \cdot t(n)})$$

3 The cost of eliminating nondeterminism

3.1 time

the trivial simulation with exponential overhead

Lemma 2. *Let $t(n)$ be time constructible. Then*

$$NTIME(t(n)) \subseteq \bigcup_c TIME(2^{c \cdot t(n)})$$

notation: we can abbreviate

$$TIME(2^{O(t(n))}) = \bigcup_c TIME(2^{c \cdot t(n)})$$

and reformulate

$$NTIME(t(n)) \subseteq TIME(2^{O(t(n))})$$

we also define

$$EXPTIME = \bigcup_c TIME(2^{c \cdot n}) = TIME(2^{O(n)})$$

$$EXPSPACE = \bigcup_c SPACE(2^{c \cdot n}) = SPACE(2^{O(n)})$$

3 The cost of eliminating nondeterminism

3.1 time

the trivial simulation with exponential overhead

Lemma 2. *Let $t(n)$ be time constructible. Then*

$$NTIME(t(n)) \subseteq \bigcup_c TIME(2^{c \cdot t(n)})$$

notation: we can abbreviate

$$TIME(2^{O(t(n))}) = \bigcup_c TIME(2^{c \cdot t(n)})$$

and reformulate

$$NTIME(t(n)) \subseteq TIME(2^{O(t(n))})$$

we also define

$$EXPTIME = \bigcup_c TIME(2^{c \cdot n}) = TIME(2^{O(n)})$$

$$EXPSPACE = \bigcup_c SPACE(2^{c \cdot n}) = SPACE(2^{O(n)})$$

proof of lemma: Let

$$M = (Z, \Sigma, \delta, z_0, Z_A)$$

be a nondeterministic k-tape TM.

oracle sequences: Let

$$g = \max\{|\delta(z,a)| \ : \ z \in Z \ , \ a \in \Sigma\}$$

the largest number of choices of M in any move. Then the nondeterministic choices of M during t steps can be specified as an *oracle sequence*

$$q[1:t] \in [1:g]^t$$

of length t where for all i sequence element $q(i)$ specifies M's choice in step i.

3 The cost of eliminating nondeterminism

3.1 time

the trivial simulation with exponential overhead

Lemma 2. *Let $t(n)$ be time constructible. Then*

$$NTIME(t(n)) \subseteq \bigcup_c TIME(2^{c \cdot t(n)})$$

notation: we can abbreviate

$$TIME(2^{O(t(n))}) = \bigcup_c TIME(2^{c \cdot t(n)})$$

and reformulate

$$NTIME(t(n)) \subseteq TIME(2^{O(t(n))})$$

we also define

$$EXPTIME = \bigcup_c TIME(2^{c \cdot n}) = TIME(2^{O(n)})$$

$$EXPSPACE = \bigcup_c SPACE(2^{c \cdot n}) = SPACE(2^{O(n)})$$

proof of lemma: Let

$$M = (Z, \Sigma, \delta, z_0, Z_A)$$

be a nondeterministic k-tape TM.

oracle sequences: Let

$$g = \max\{|\delta(z,a)| \ : \ z \in Z \ , \ a \in \Sigma\}$$

the largest number of choices of M in any move. Then the nondeterministic choices of M during t steps can be specified as an *oracle sequence*

$$q[1:t] \in [1:g]^t$$

of length t where for all i sequence element $q(i)$ specifies M's choice in step i.

simulation of M: by deterministic $(k+1)$-tape TM M' on input w with $|w| = n$:

- compute $t(n)$.
- in $g^{t(n)}$ rounds enumerate the possible oracle sequences $q^{(r)} \in [1:g]^t$
- in each round r run M with the oracle sequence $q^{(r)}$
- if M accepts in any round r accept, otherwise reject.

3 The cost of eliminating nondeterminism

3.1 time

the trivial simulation with exponential overhead

Lemma 2. *Let $t(n)$ be time constructible. Then*

$$NTIME(t(n)) \subseteq \bigcup_c TIME(2^{c\cdot t(n)})$$

notation: we can abbreviate

$$TIME(2^{O(t(n))}) = \bigcup_c TIME(2^{c\cdot t(n)})$$

and reformulate

$$NTIME(t(n)) \subseteq TIME(2^{O(t(n))})$$

we also define

$$EXPTIME = \bigcup_c TIME(2^{c\cdot n}) = TIME(2^{O(n)})$$

$$EXPSPACE = \bigcup_c SPACE(2^{c\cdot n}) = SPACE(2^{O(n)})$$

proof of lemma: Let

$$M = (Z, \Sigma, \delta, z_0, Z_A)$$

be a nondeterministic k-tape TM.

oracle sequences: Let

$$g = \max\{|\delta(z,a)| \ : \ z \in Z \ , \ a \in \Sigma\}$$

the largest number of choices of M in any move. Then the nondeterministic choices of M during t steps can be specified as an *oracle sequence*

$$q[1:t] \in [1:g]^t$$

of length t where for all i sequence element $q(i)$ specifies M's choice in step i.

simulation of M: by deterministic $(k+1)$-tape TM M' on input w with $|w| = n$:

- compute $t(n)$.
- in $g^{t(n)}$ rounds enumerate the possible oracle sequences $q^{(r)} \in [1:g]^t$
- in each round r run M with the oracle sequence $q^{(r)}$
- if M accepts in any round r accept, otherwise reject.

time $T(n)$ used: for large n

$$\begin{aligned} T(n) &\leq t(n) + g^{t(n)} \cdot A \cdot t(n) \\ &\leq A \cdot g^{2t(n)} \\ &= A \cdot 2^{(2\log g)\cdot t(n)} \end{aligned}$$

exercises

1. is the hypothesis that $t(n)$ is time constructible necessary? Hint: try $t = 2, 4, 8, 16, , \ldots$

exercises

1. is the hypothesis that $t(n)$ is time constructible necessary? Hint: try $t = 2,4,8,16,,\ldots$

2. We now know

$$TIME(t(n)) \subseteq NTIME(t(n)) \subseteq TIME(2^{O(n)})$$

Prove for time constructible $t(n)$, that we cannot have equality on both sides, i.e. that for time constructible $t(n)$

$$TIME(t(n)) = NTIME(t(n)) = TIME(2^{O(n)})$$

is impossible

exercises

1. is the hypothesis that $t(n)$ is time constructible necessary? Hint: try $t = 2,4,8,16,,\ldots$

2. We now know

$$TIME(t(n)) \subseteq NTIME(t(n)) \subseteq TIME(2^{O(n)})$$

   Prove for time constructible $t(n)$, that we cannot have equality on both sides, i.e. that for time constructible $t(n)$

$$TIME(t(n)) = NTIME(t(n)) = TIME(2^{O(n)})$$

   is impossible

3. do you think

$$TIME(t(n)) = NTIME(t(n)) = TIME(2^{O(n)})$$

   might be possible for functions $t(n)$ which are not time constructible? Hint: google Allan Borodin

exercises

1. is the hypothesis that $t(n)$ is time constructible necessary? Hint: try $t = 2, 4, 8, 16, , \ldots$

2. We now know

$$TIME(t(n)) \subseteq NTIME(t(n)) \subseteq TIME(2^{O(n)})$$

Prove for time constructible $t(n)$, that we cannot have equality on both sides, i.e. that for time constructible $t(n)$

$$TIME(t(n)) = NTIME(t(n)) = TIME(2^{O(n)})$$

is impossible

3. do you think

$$TIME(t(n)) = NTIME(t(n)) = TIME(2^{O(n)})$$

might be possible for functions $t(n)$ which are not time constructible? Hint: google Allan Borodin

4. show $NTIME(t(n)) \subseteq SPACE(t(n))$

3.2 space

Savitch's theorem: overhead is only quadratic.

Lemma 3. *For space constructible $s(n)$:*

$$NSPACE(s(n)) \subseteq SPACE(s^2(n))$$

3.2 space

Savitch's theorem: overhead is only quadratic.

Lemma 3. *For space constructible $s(n)$:*

$$NSPACE(s(n)) \subseteq SPACE(s^2(n))$$

Let

$$M = (Z, \Sigma, \delta, z_0, Z_A)$$

be a nondeterministic k-tape TM. W.l.o.g. M has 1 tape and $Z_A = \{z_a\}$. Modify M such that it accepts with empty tape, the head at the original position. Accepting computations of M started with input w on space s and time t

- start from configuration
$$k_0 = B^s z_0 w B^{s-|w|}$$
- end in a configuration
$$k_t = B^s z_a B^s$$
- never exceed the cells belonging to k_0
- obey
$$t \leq 2s \cdot |Z| \cdot |\Sigma|^{2s}$$

otherwise there is a shorter accepting computation

3.2 space

Savitch's theorem: overhead is only quadratic.

Lemma 3. *For space constructible $s(n)$:*

$$NSPACE(s(n)) \subseteq SPACE(s^2(n))$$

Let

$$M = (Z, \Sigma, \delta, z_0, Z_A)$$

be a nondeterministic k-tape TM. W.l.o.g. M has 1 tape and $Z_A = \{z_a\}$. Modify M such that it accepts with empty tape, the head at the original position. Accepting computations of M started with input w on space s and time t

- start from configuration
$$k_0 = B^s z_0 w B^{s-|w|}$$
- end in a configuration
$$k_t = B^s z_a B^s$$
- never exceed the cells belonging to k_0
- obey
$$t \leq 2s \cdot |Z| \cdot |\Sigma|^{2s}$$

otherwise there is a shorter accepting computation

For configurations k, k' of M define

$$k \vdash_s^t k'$$

iff M can get from configuration k to configuration k' in at most t steps and never exceeds the cells belonging to k. If t is a power of two we obviously have

$$k \vdash_s^t k' \leftrightarrow \exists k''.(k \vdash_s^{t/2} k'' \wedge k'' \vdash_s^{t/2} k')$$

For configurations k, k' of M define

$$k \vdash_s^t k'$$

iff M can get from configuration k to configuration k' in at most t steps and never exceeds the cells belonging to k. If t is a power of two we obviously have

$$\boxed{k \vdash_s^t k' \leftrightarrow \exists k''.(k \vdash_s^{t/2} k'' \ \wedge \ k'' \vdash_s^{t/2} k')}$$

simulation by M' executes function $test(k, k', s, bin(t))$ which recursively checks $k \vdash_s^t k'$.

- initialization with input w:

  with $n = |w|$ and

  $$s = s(n) \ , \ t' = 2s \cdot |Z| \cdot |\Sigma|^{2s} \ , \ t = \min\{2^j \geq t' \ : \ j \in \mathbb{N}\}$$

  and

  $$k = B^s z_0 w B^{s-n} \ , \ k' = B^s z_a B^s$$

  call

  $$test(k, k', s, t)$$

  by generating 'stack frame'

  $$k\$k'\$bin(t)\$$$

For configurations k, k' of M define

$$k \vdash_s^t k'$$

iff M can get from configuration k to configuration k' in at most t steps and never exceeds the cells belonging to k. If t is a power of two we obviously have

$$\boxed{k \vdash_s^t k' \leftrightarrow \exists k''.(k \vdash_s^{t/2} k'' \ \wedge \ k'' \vdash_s^{t/2} k')}$$

simulation by M' executes function $test(k, k', s, bin(t))$ which recursively checks $k \vdash_s^t k'$.

- initialization with input w:

  with $n = |w|$ and

  $$s = s(n) \ , \ t' = 2s \cdot |Z| \cdot |\Sigma|^{2s} \ , \ t = \min\{2^j \geq t' \ : \ j \in \mathbb{N}\}$$

  and

  $$k = B^s z_0 w B^{s-n} \ , \ k' = B^s z_a B^s$$

  call

  $$test(k, k', s, t)$$

  by generating 'stack frame'

  $$k\$k'\$bin(t)\$$$

- recursion step: for $t > 1$ and top (rightmost) frame

  $$k\$k'\$bin(t)\$$$

  1. enumerate in lexicographical order all configurations k'' on the cells of the next stack frame.
  2. for each k'' call

     $$test(k, k'', t/2)$$

     with a frame

     $$0\$k\$k''\$bin(t/2)\$$$

     where the leading 0 indicates that this is the first of 2 recursive calls for k''.
  3. If this fails try next k''
  4. If it succeeds call

     $$test(k'', k', t/2)$$

     with a frame

     $$1\$k''\$k'\$bin(t/2)\$$$

  5. if this fails goto 2 and try next k''.
  6. it succeeds return *success*.
  7. if all k'' are exhausted without *success* return $fail$.

- end of recursion: for $t = 1$ this one performs the trivial check

  $$k \vdash k'$$

bounding space $S(n)$ of the simulation

- computing $bin(s(n))$: space $O(s(n))$ as $s(n)$ is space constructible
- computing initial $bin(t)$: space

$$\lceil \log(2s \cdot |Z| \cdot |\Sigma|^{2s}) \rceil = O(s(n))$$

- enumerating configurations k'' and generating a stack frame:

$$O(s(n))$$

- recursion depth:

$$\log t = O(s(n))$$

- space bound

$$S(n) = O(s^2(n))$$

bounding space $S(n)$ of the simulation

- computing $bin(s(n))$: space $O(s(n))$ as $s(n)$ is space constructible
- computing initial $bin(t)$: space

$$\lceil \log(2s \cdot |Z| \cdot |\Sigma|^{2s}) \rceil = O(s(n))$$

- enumerating configurations k'' and generating a stack frame:

$$O(s(n))$$

- recursion depth:

$$\log t = O(s(n))$$

- space bound

$$S(n) = O(s^2(n))$$

exercise: is the hypothesis that $s(n)$ is space constructible necessary?

4 Linear bounded automata (LBA's) and context sensitive languages

def: linear bounded automaton (LBA) A linear bounded automaton *LBA* is an $O(n)$-space bounded nondeterministic Turing machine.

$$NSPACE(n) = \{L \ : \ L \text{ is accepted by an LBA}\}$$

4.1 LBA's accept all context sensitive languages

Lemma 4. *Let L be a context sensitive language, then*

$$L \in NSPACE(n)$$

4 Linear bounded automata (LBA's) and context sensitive languages

def: linear bounded automaton (LBA) A linear bounded automaton *LBA* is an $O(n)$-space bounded nondeterministic Turing machine.

$$NSPACE(n) = \{L \ : \ L \text{ is accepted by an LBA}\}$$

4.1 LBA's accept all context sensitive languages

Lemma 4. *Let L be a context sensitive language, then*

$$L \in NSPACE(n)$$

- let $L = L(G)$ for cs grammar $G = (N,T,P,S)$. We construct an LBA M accepting L as follows
- given input w nondeterministically reduce w by applying productions of P backwards
- this never exceeds the room originally taken by w, as left hand sides of productions cannot be longer than right hand sides.
- accept if reduction to S succeeds

4 Linear bounded automata (LBA's) and context sensitive languages

def: linear bounded automaton (LBA) A linear bounded automaton *LBA* is an $O(n)$-space bounded nondeterministic Turing machine.

$$NSPACE(n) = \{L \ : \ L \text{ is accepted by an LBA}\}$$

4.1 LBA's accept all context sensitive languages

Lemma 4. *Let L be a context sensitive language, then*

$$L \in NSPACE(n)$$

- let $L = L(G)$ for cs grammar $G = (N,T,P,S)$. We construct an LBA M accepting L as follows
- given input w nondeterministically reduce w by applying productions of P backwards
- this never exceeds the room originally taken by w, as left hand sides of productions cannot be longer than right hand sides.
- accept if reduction to S succeeds

4.2 LBA's accept exactly the context sensitive languages

Lemma 5. *Let L be a context sensitive language if and only if*

$$L \in NSPACE(n)$$

$\leftarrow$: let L be accepted by LBA M. W.l.o.g. M has 1 tape and M is $s(n)$-space bounded with

$$s(n) \leq c \cdot n \quad \text{for all } n \geq n_0$$

With $r(n) = s(n)/n$ set

$$c' = \max\{r(n) \ : \ n < n_0\}$$

then

$$s(n) = r(n) \cdot n \leq c' \cdot n \quad \text{for } n < n_0$$

and with $C = \max\{c, c'\}$

$$s(n) \leq C \cdot n \quad \text{for all } n$$

4 Linear bounded automata (LBA's) and context sensitive languages

def: linear bounded automaton (LBA) A linear bounded automaton *LBA* is an $O(n)$-space bounded nondeterministic Turing machine.

$$NSPACE(n) = \{L \ : \ L \text{ is accepted by an LBA}\}$$

4.1 LBA's accept all context sensitive languages

Lemma 4. *Let L be a context sensitive language, then*

$$L \in NSPACE(n)$$

- let $L = L(G)$ for cs grammar $G = (N,T,P,S)$. We construct an LBA M accepting L as follows
- given input w nondeterministically reduce w by applying productions of P backwards
- this never exceeds the room originally taken by w, as left hand sides of productions cannot be longer than right hand sides.
- accept if reduction to S succeeds

4.2 LBA's accept exactly the context sensitive languages

Lemma 5. *Let L be a context sensitive language if and only if*

$$L \in NSPACE(n)$$

$\leftarrow$: let L be accepted by LBA M. W.l.o.g. M has 1 tape and M is $s(n)$-space bounded with

$$s(n) \leq c \cdot n \quad \text{for all } n \geq n_0$$

With $r(n) = s(n)/n$ set

$$c' = \max\{r(n) \ : \ n < n_0\}$$

then

$$s(n) = r(n) \cdot n \leq c' \cdot n \quad \text{for } n < n_0$$

and with $C = \max\{c, c'\}$

$$s(n) \leq C \cdot n \quad \text{for all } n$$

- modify M as follows:
- apply tape compression by factor C. This gives machine M' which is n-space bounded, but might move to the left of the original input.

4 Linear bounded automata (LBA's) and context sitive languages

4.2 LBA's accept exactly the context sensitive languages

Lemma 5. *Let L be a context sensitive language if and only if*

$$L \in NSPACE(n)$$

$\leftarrow$: let L be accepted by LBA M. W.l.o.g. M has 1 tape and M is $s(n)$-sp bounded with

$$s(n) \leq c \cdot n \quad \text{for all } n \geq n_0$$

With $r(n) = s(n)/n$ set

$$c' = \max\{r(n) \ : \ n < n_0\}$$

then

$$s(n) = r(n) \cdot n \leq c' \cdot n \quad \text{for } n < n_0$$

and with $C = \max\{c, c'\}$

$$s(n) \leq C \cdot n \quad \text{for all } n$$

- modify M as follows:
- apply tape compression by factor C. This gives machine M' which is n-space bounded, but might move to the left of the original input.
- modify M as follows:
- apply tape compression by factor C. This gives machine M' which is n-space bounded, but might move to the left of the original input.
- Modify to machine M''. With input w use 3 tracks.

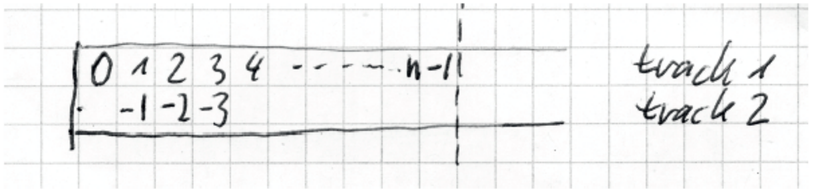

Figure 1: Folding a tape on 2 tracks. Only the portion right of the original head position is visited

1. track 1: first copy input w there
2. track 2: contains the visited portion of the tape of M' left of the original head position folded backwards.
3. track 3: mark folding point on cell 0

Thus the tape of M' is folded on tracks 1 and 2. On track 2 moves R/L of M' are simulated by moves L/R of M''.

4 Linear bounded automata (LBA's) and context sitive languages

4.2 LBA's accept exactly the context sensitive languages

Lemma 5. *Let L be a context sensitive language if and only if*

$$L \in NSPACE(n)$$

$\leftarrow$: let L be accepted by LBA M. W.l.o.g. M has 1 tape and M is $s(n)$-sp bounded with

$$s(n) \leq c \cdot n \quad \text{for all } n \geq n_0$$

With $r(n) = s(n)/n$ set

$$c' = \max\{r(n) \ : \ n < n_0\}$$

then

$$s(n) = r(n) \cdot n \leq c' \cdot n \quad \text{for } n < n_0$$

and with $C = \max\{c, c'\}$

$$s(n) \leq C \cdot n \quad \text{for all } n$$

- modify M as follows:
- apply tape compression by factor C. This gives machine M' which is n-space bounded, but might move to the left of the original input.
- modify M as follows:
- apply tape compression by factor C. This gives machine M' which is n-space bounded, but might move to the left of the original input.
- Modify to machine M''. With input w use 3 tracks.

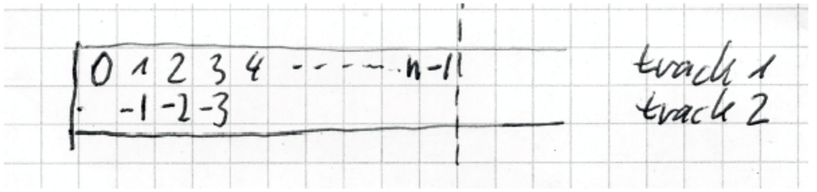

Figure 1: Folding a tape on 2 tracks. Only the portion right of the original head position is visited

1. track 1: first copy input w there
2. track 2: contains the visited portion of the tape of M' left of the original head position folded backwards.
3. track 3: mark folding point on cell 0

Thus the tape of M' is folded on tracks 1 and 2. On track 2 moves R/L of M' are simulated by moves L/R of M''.

- M'' works only on the space occupied by the original input.
- now simulate M'' by a cs grammar as in the simulation of Turing machines by type-0 grammars.

4.3 Completing the Chomsky hierarchy

Lemma 6. *There exist type-0 languages which are not context sensitive.*

Proof. Exercise. Hint: use (among other things) the space hierarchy theorem □

5 Polynomially bounded resources

5.1 a few definitions

polynomial time

$$P = \bigcup_c TIME(n^c)$$

- Languages decidable by *feasible* computations

5 Polynomially bounded resources

5.1 a few definitions

polynomial time

$$P = \bigcup_c TIME(n^c)$$

- Languages decidable by *feasible* computations

- within limits independent of the model of computation: 1-tape TMs, k-tape TMS, log-RAMs simulate each other with polynomial slowdown.

5 Polynomially bounded resources

5.1 a few definitions

polynomial time

$$P = \bigcup_c TIME(n^c)$$

- Languages decidable by *feasible* computations

- within limits independent of the model of computation: 1-tape TMs, k-tape TMS, log-RAMs simulate each other with polynomial slowdown.

- $TIME(n^{2437}) \subseteq P$ is feasible only with a LOT of good will.

  exercise: why does

  $$TIME(n^{2437}) \subsetneq P$$

  hold?

polynomial space

$$PSPACE = \bigcup_c SPACE(n^c)$$

exercise:

$$PSPACE = \bigcup_c NSPACE(n^c)$$

5 Polynomially bounded resources

5.1 a few definitions

polynomial time

$$P = \bigcup_c TIME(n^c)$$

- Languages decidable by *feasible* computations

nondeterministic polynomial time

$$NP = \bigcup_c NTIME(n^c)$$

- within limits independent of the model of computation: 1-tape TMs, k-tape TMS, log-RAMs simulate each other with polynomial slowdown.

- $TIME(n^{2437}) \subseteq P$ is feasible only with a LOT of good will.

  exercise: why does

  $$TIME(n^{2437}) \subsetneq P$$

  hold?

polynomial space

$$PSPACE = \bigcup_c SPACE(n^c)$$

exercise:

$$PSPACE = \bigcup_c NSPACE(n^c)$$

5 Polynomially bounded resources

5.1 a few definitions

polynomial time

$$P = \bigcup_c TIME(n^c)$$

- Languages decidable by *feasible* computations

nondeterministic polynomial time

$$NP = \bigcup_c NTIME(n^c)$$

- within limits independent of the model of computation: 1-tape TMs, k-tape TMS, log-RAMs simulate each other with polynomial slowdown.
- $TIME(n^{2437}) \subseteq P$ is feasible only with a LOT of good will.

  exercise: why does

  $$TIME(n^{2437}) \subsetneq P$$

  hold?
- who cares?
- everybody and there is a reason
- $P \subseteq NP$ trivial

polynomial space

$$PSPACE = \bigcup_c SPACE(n^c)$$

exercise:

$$PSPACE = \bigcup_c NSPACE(n^c)$$

5 Polynomially bounded resources

5.1 a few definitions

polynomial time

$$P = \bigcup_c TIME(n^c)$$

- Languages decidable by *feasible* computations

nondeterministic polynomial time

$$NP = \bigcup_c NTIME(n^c)$$

- within limits independent of the model of computation: 1-tape TMs, k-tape TMS, log-RAMs simulate each other with polynomial slowdown.
- who cares?
- everybody and there is a reason
- $TIME(n^{2437}) \subseteq P$ is feasible only with a LOT of good will.

  exercise: why does

  $$TIME(n^{2437}) \subsetneq P$$

  hold?
- $P \subseteq NP$ trivial
- the most famous problem of mathematics and (at least of theoretical) computer science:

  $$P = NP?$$
- we will see why

polynomial space

$$PSPACE = \bigcup_c SPACE(n^c)$$

exercise:

$$PSPACE = \bigcup_c NSPACE(n^c)$$

6 Padding

a technique to separate complexity classes Practically never treated in textbooks. Used (introduced?) 1972 by Ron Book. Not appreciated enough.

6 Padding

a technique to separate complexity classes Practically never treated in textbooks. Used (introduced?) 1972 by Ron Book. Not appreciated enough.

6.1 Refining the time hierarchy

Lemma 7. *Let $t(n)$ be time constructible. Then*

$$TIME(n) \subsetneq TIME(n^{1.5})$$

- assume

  $$TIME(n) = TIME(n^{1.5})$$

  Let

  $$L \in TIME(n^{2.25})$$

  accepted by $O(n^{2.25})$-time bounded TM M.

6 Padding

a technique to separate complexity classes Practically never treated in textbooks. Used (introduced?) 1972 by Ron Book. Not appreciated enough.

6.1 Refining the time hierarchy

Lemma 7. *Let $t(n)$ be time constructible. Then*

$$TIME(n) \subsetneq TIME(n^{1.5})$$

- assume

  $$TIME(n) = TIME(n^{1.5})$$

  Let

  $$L \in TIME(n^{2.25})$$

  accepted by $O(n^{2.25})$-time bounded TM M.

- with new 'padding' symbol X define

  $$L' = \{wX^c \ : \ w \in L \ , \ |wX^c| = \lceil |w|^{1.5} \rceil\}$$

6 Padding

a technique to separate complexity classes Practically never treated in textbooks. Used (introduced?) 1972 by Ron Book. Not appreciated enough.

6.1 Refining the time hierarchy

Lemma 7. *Let $t(n)$ be time constructible. Then*

$$TIME(n) \subsetneq TIME(n^{1.5})$$

- assume

  $$TIME(n) = TIME(n^{1.5})$$

  Let

  $$L \in TIME(n^{2.25})$$

  accepted by $O(n^{2.25})$-time bounded TM M.

- with new 'padding' symbol X define

  $$L' = \{wX^c \ : \ w \in L \ , \ |wX^c| = \lceil |w|^{1.5} \rceil\}$$

- accept L' by M'. With input wX^c

  1. check $|wX^c| = \lceil |w|^{1.5} \rceil$. If not reject. Otherwise
  2. erase X^c and run M on input w. Accept iff M accepts.

6 Padding

a technique to separate complexity classes Practically never treated in textbooks. Used (introduced?) 1972 by Ron Book. Not appreciated enough.

6.1 Refining the time hierarchy

Lemma 7. *Let $t(n)$ be time constructible. Then*

$$TIME(n) \subsetneq TIME(n^{1.5})$$

- assume

$$TIME(n) = TIME(n^{1.5})$$

Let

$$L \in TIME(n^{2.25})$$

accepted by $O(n^{2.25})$-time bounded TM M.

- with new 'padding' symbol X define

$$L' = \{wX^c \ : \ w \in L \ , \ |wX^c| = \lceil |w|^{1.5} \rceil\}$$

- accept L' by M'. With input wX^c
  1. check $|wX^c| = \lceil |w|^{1.5} \rceil$. If not reject. Otherwise
  2. erase X^c and run M on input w. Accept iff M accepts.

- M' started with input of length $n = \lceil m^{1.5} \rceil$ makes

$$O(m^{2.25}) = O(n^{1.5})$$

steps. Thus by asumption

$$L' \in TIME(n^{1.5}) = TIME(n)$$

6 Padding

a technique to separate complexity classes Practically never treated in textbooks. Used (introduced?) 1972 by Ron Book. Not appreciated enough.

6.1 Refining the time hierarchy

Lemma 7. *Let $t(n)$ be time constructible. Then*

$$TIME(n) \subsetneq TIME(n^{1.5})$$

- assume

$$TIME(n) = TIME(n^{1.5})$$

Let

$$L \in TIME(n^{2.25})$$

accepted by $O(n^{2.25})$-time bounded TM M.

- with new 'padding' symbol X define

$$L' = \{wX^c \ : \ w \in L \ , \ |wX^c| = \lceil |w|^{1.5} \rceil\}$$

- accept L' by M'. With input wX^c
  1. check $|wX^c| = \lceil |w|^{1.5} \rceil$. If not reject. Otherwise
  2. erase X^c and run M on input w. Accept iff M accepts.

- M' started with input of length $n = \lceil m^{1.5} \rceil$ makes

$$O(m^{2.25}) = O(n^{1.5})$$

steps. Thus by asumption

$$L' \in TIME(n^{1.5}) = TIME(n)$$

- accept L by M'' as follows.
  1. pad w to wX^c with $|wX^c| = \lceil |w|^{1.5} \rceil$.
  2. behave as $O(n)$-time bounded acceptor for L' with this input.

6 Padding

a technique to separate complexity classes Practically never treated in textbooks. Used (introduced?) 1972 by Ron Book. Not appreciated enough.

6.1 Refining the time hierarchy

Lemma 7. *Let $t(n)$ be time constructible. Then*

$$TIME(n) \subsetneq TIME(n^{1.5})$$

- assume
$$TIME(n) = TIME(n^{1.5})$$
Let
$$L \in TIME(n^{2.25})$$
accepted by $O(n^{2.25})$-time bounded TM M.

- with new 'padding' symbol X define
$$L' = \{wX^c \ : \ w \in L \ , \ |wX^c| = \lceil |w|^{1.5} \rceil\}$$

- accept L' by M'. With input wX^c
  1. check $|wX^c| = \lceil |w|^{1.5} \rceil$. If not reject. Otherwise
  2. erase X^c and run M on input w. Accept iff M accepts.

- M' started with input of length $n = \lceil m^{1.5} \rceil$ makes
$$O(m^{2.25}) = O(n^{1.5})$$
steps. Thus by asumption
$$L' \in TIME(n^{1.5}) = TIME(n)$$

- accept L by M'' as follows.
  1. pad w to wX^c with $|wX^c| = \lceil |w|^{1.5} \rceil$.
  2. behave as $O(n)$-time bounded acceptor for L' with this input.

- M'' is $O(n^{1.5})$-time bounded. Thus again by assumption
$$L \in TIME(n^{1.5}) = TIME(n)$$

- as $L \in TIME(n^{2.25})$ was arbitrary
$$TIME(n) = TIME(n^{2.25})$$
contradicting the time hierarchy theorem.

6.2 Separating linear space from P

Lemma 8. *If for* $r \in \mathbb{N}$

$$SPACE(n^r) \subseteq P$$

then

$$PSPACE \subseteq P \subseteq PSPACE$$

i.e.

$$PSPACE = P$$

6.2 Separating linear space from P

Lemma 8. *If for* $r \in \mathbb{N}$

$$SPACE(n^r) \subseteq P$$

then

$$PSPACE \subseteq P \subseteq PSPACE$$

i.e.

$$PSPACE = P$$

- Let $L \in PSPACE$, i.e.

  $$L \in SPACE(n^s)$$

  for some $s > r$ and let M be an $O(n^s)$ space bounded acceptor for L.

6.2 Separating linear space from P

Lemma 8. *If for* $r \in \mathbb{N}$

$$SPACE(n^r) \subseteq P$$

then

$$PSPACE \subseteq P \subseteq PSPACE$$

i.e.

$$PSPACE = P$$

- Let $L \in PSPACE$, i.e.

  $$L \in SPACE(n^s)$$

  for some $s > r$ and let M be an $O(n^s)$ space bounded acceptor for L.

- with new padding symbol X define

  $$L' = \{wX^c \ : \ |wX^c| = |w|^s\}$$

  Accept L' by $O(n)$-space bounded TM M which first checks the format and if fitting erases the padding symbols and behaves like M. Thus by hypothesis

  $$L' \in SPACE(n) \subseteq SPACE(n^r) \subseteq P$$

6.2 Separating linear space from P

Lemma 8. *If for* $r \in \mathbb{N}$

$$SPACE(n^r) \subseteq P$$

then

$$PSPACE \subseteq P \subseteq PSPACE$$

i.e.

$$PSPACE = P$$

- Let $L \in PSPACE$, i.e.

  $$L \in SPACE(n^s)$$

  for some $s > r$ and let M be an $O(n^s)$ space bounded acceptor for L.

- with new padding symbol X define

  $$L' = \{wX^c \ : \ |wX^c| = |w|^s\}$$

  Accept L' by $O(n)$-space bounded TM M which first checks the format and if fitting erases the padding symbols and behaves like M. Thus by hypothesis

  $$L' \in SPACE(n) \subseteq SPACE(n^r) \subseteq P$$

Lemma 9. *For no numbers* $r, s \in \mathbb{N}$ *with* $1 \leq r \leq s$ *holds*

$$SPACE(n^r) \subseteq P \subseteq SPACE(n^s)$$

6.2 Separating linear space from P

Lemma 8. *If for* $r \in \mathbb{N}$

$$SPACE(n^r) \subseteq P$$

then

$$PSPACE \subseteq P \subseteq PSPACE$$

i.e.

$$PSPACE = P$$

- Let $L \in PSPACE$, i.e.

  $$L \in SPACE(n^s)$$

  for some $s > r$ and let M be an $O(n^s)$ space bounded acceptor for L.

- with new padding symbol X define

  $$L' = \{wX^c \ : \ |wX^c| = |w|^s\}$$

  Accept L' by $O(n)$-space bounded TM M which first checks the format and if fitting erases the padding symbols and behaves like M. Thus by hypothesis

  $$L' \in SPACE(n) \subseteq SPACE(n^r) \subseteq P$$

Lemma 9. *For no numbers* $r, s \in \mathbb{N}$ *with* $1 \leq r \leq s$ *holds*

$$SPACE(n^r) \subseteq P \subseteq SPACE(n^s)$$

Proof. assume true. By lemma 8

$$SPACE(n^r) \subseteq PSPACE = P \subseteq SPACE(n^s)$$

contradicting the space hierarchy theorem. $PSPACE \setminus SPACE(n^s) \neq \emptyset$

6.2 Separating linear space from P

Lemma 8. *If for* $r \in \mathbb{N}$

$$SPACE(n^r) \subseteq P$$

then

$$PSPACE \subseteq P \subseteq PSPACE$$

i.e.

$$PSPACE = P$$

- Let $L \in PSPACE$, i.e.

  $$L \in SPACE(n^s)$$

  for some $s > r$ and let M be an $O(n^s)$ space bounded acceptor for L.

- with new padding symbol X define

  $$L' = \{wX^c \ : \ |wX^c| = |w|^s\}$$

  Accept L' by $O(n)$-space bounded TM M which first checks the format and if fitting erases the padding symbols and behaves like M. Thus by hypothesis

  $$L' \in SPACE(n) \subseteq SPACE(n^r) \subseteq P$$

Lemma 9. *For no numbers* $r, s \in \mathbb{N}$ *with* $1 \leq r \leq s$ *holds*

$$SPACE(n^r) \subseteq P \subseteq SPACE(n^s)$$

Proof. assume true. By lemma 8

$$SPACE(n^r) \subseteq PSPACE = P \subseteq SPACE(n^s)$$

contradicting the space hierarchy theorem. $PSPACE \setminus SPACE(n^s) \neq \emptyset$

Lemma 10. *For all* r *holds* $P \neq SPACE(n^r)$

Proof. Set $s = r$ in lemma 10

6.2 Separating linear space from P

Lemma 8. *If for* $r \in \mathbb{N}$

$$SPACE(n^r) \subseteq P$$

then

$$PSPACE \subseteq P \subseteq PSPACE$$

i.e.

$$PSPACE = P$$

- Let $L \in PSPACE$, i.e.

  $$L \in SPACE(n^s)$$

  for some $s > r$ and let M be an $O(n^s)$ space bounded acceptor for L.

- with new padding symbol X define

  $$L' = \{wX^c \ : \ |wX^c| = |w|^s\}$$

  Accept L' by $O(n)$-space bounded TM M which first checks the format and if fitting erases the padding symbols and behaves like M. Thus by hypothesis

  $$L' \in SPACE(n) \subseteq SPACE(n^r) \subseteq P$$

Lemma 9. *For no numbers* $r, s \in \mathbb{N}$ *with* $1 \leq r \leq s$ *holds*

$$SPACE(n^r) \subseteq P \subseteq SPACE(n^s)$$

Proof. assume true. By lemma 8

$$SPACE(n^r) \subseteq PSPACE = P \subseteq SPACE(n^s)$$

contradicting the space hierarchy theorem. $PSPACE \setminus SPACE(n^s) \neq \emptyset$

Lemma 10. *For all* r *holds* $P \neq SPACE(n^r)$

Proof. Set $s = r$ in lemma 10

exercise: show in a similar way

1. $P \neq NSPACE(n^r)$
2. $NP \neq SPACE(n^r)$
3. $NP \neq NSPACE(n^r)$. In particular NP is not the set of context sensitive languages ($r = 1$).