Complexity and Cryptography

Thomas Zeugmann

Hokkaido University
Laboratory for Algorithmics

https://www-alg.ist.hokudai.ac.jp/~thomas/COCRB/

Lecture 8: More about Complexity Classes

More about Tape Reductions I

We ask whether or not we can improve our results obtained so far for NTMs.

Surprisingly, allowing two work tapes for NTMs is already sufficient to simulate any NTM without increasing the time complexity, i.e., we can show the following:

More about Tape Reductions I

We ask whether or not we can improve our results obtained so far for NTMs.

Surprisingly, allowing two work tapes for NTMs is already sufficient to simulate any NTM without increasing the time complexity, i.e., we can show the following:

Theorem 1 (Book, Greibach, Wegbreit (1970))

Let $f\colon \mathbb{N} \to \mathbb{N}$ *be any bounding function. Then we have*

$$\mathit{NTIME}(f(n)) = \mathit{NTime}_3(f(n)) \,.$$

More about Tape Reductions II

Proof. Let any nondeterministic k-tape TM $M = [B, Z, A]$ be given such that $k > 3$. If $k \leqslant 3$, the theorem is obvious. Now, we construct a nondeterministic 3-tape TM M^* with $L(M^*) = L(M)$ and $T_{M^*}(w) = O(T_M(w))$ for all w.

More about Tape Reductions II

Proof. Let any nondeterministic k-tape TM $M = [B, Z, A]$ be given such that $k > 3$. If $k \leqslant 3$, the theorem is obvious. Now, we construct a nondeterministic 3-tape TM M^* with $L(M^*) = L(M)$ and $T_{M^*}(w) = O(T_M(w))$ for all w.

On its first work tape M^* is using the tape alphabet B and on its second work tape the tape alphabet $Z \times B^k$.

On input w, the machine M^* works as follows:

More about Tape Reductions III

(1) M^* writes nondeterministically an arbitrary finite number of symbols on its second work tape. Note that these symbols are from $Z \times B^k$, i.e., $(k+1)$-tuples.

More about Tape Reductions III

(1) M^* writes nondeterministically an arbitrary finite number of symbols on its second work tape. Note that these symbols are from $Z \times B^k$, i.e., $(k+1)$-tuples.

(2) The ith symbol $(z^i, b^1, \ldots, b^k)$ written on the second work tape is interpreted as the actual situation machine M is in its ith step, i.e., it is assumed to be in state z^i and observing $b^1, \ldots, b^k$ on its input tape and its $k-1$ work tapes.

More about Tape Reductions III

(1) M^* writes nondeterministically an arbitrary finite number of symbols on its second work tape. Note that these symbols are from $Z \times B^k$, i.e., $(k+1)$-tuples.

(2) The ith symbol $(z^i, b^1, \ldots, b^k)$ written on the second work tape is interpreted as the actual situation machine M is in its ith step, i.e., it is assumed to be in state z^i and observing $b^1, \ldots, b^k$ on its input tape and its $k-1$ work tapes.

(3) Using the Turing table of M, the machine M^* checks deterministically whether or not the sequence of symbols on its second work tape is an accepting computation of M on input w. If it is, M^* accepts the input w, otherwise the input w is rejected.

The check is done in $k-1$ stages. In stage i, $1 \leqslant i \leqslant k-1$, the ith work tape of M is completely documented on the first work tape of M^*.

More about Tape Reductions IV

Hence, M^* can find an accepting computation path in time $k \cdot T_M(w) = O(T_M(w))$ if and only if $w \in L(M)$.

Finally, the theorem follows by our constant factor speed-up Theorem from Lecture 6 (see Theorem 6.3). ■

More about Tape Reductions V

Dealing with nondeterministic space complexity does not require any new idea. Looking at the proof of Theorem 7.1, we immediately realize that the construction performed there can also be applied in the nondeterministic case. Since this tape reduction does not increase the amount of space needed, we get the following corollary:

More about Tape Reductions V

Dealing with nondeterministic space complexity does not require any new idea. Looking at the proof of Theorem 7.1, we immediately realize that the construction performed there can also be applied in the nondeterministic case. Since this tape reduction does not increase the amount of space needed, we get the following corollary:

Corollary 1

Let $f\colon \mathbb{N} \to \mathbb{N}$ *be any bounding function. Then we have*

$$\mathit{NSPACE}(f(n)) = \mathit{NSpace}_2(f(n)) .$$

More about Tape Reductions VI

Theorem 2

Assuming an appropriate enumeration of all NTMs the following holds:

(1) *There is a universal NTM* V *for all NTMs* $(M_i)_{i \in \mathbb{N}}$ *such that*

$$L(V) = \{bin(i) * w \mid w \in L(M_i),\ i \in \mathbb{N}\}\,,$$

and for every $i \in \mathbb{N}$ *there is a constant* c_i *such that for all* $w \in \Sigma^*$ *the condition* $T_V(|bin(i) * w|) < c_i \cdot T_{M_i}(|w|)$ *is fulfilled.*

(2) *There is a universal NTM* V *for all NTMs* $(M_i)_{i \in \mathbb{N}}$ *such that*

$$L(V) = \{bin(i) * w \mid w \in L(M_i),\ i \in \mathbb{N}\}\,,$$

and for every $i \in \mathbb{N}$ *there is a constant* c_i *such that for all* $w \in \Sigma^*$ *the condition* $S_V(|bin(i) * w|) < c_i \cdot S_{M_i}(|w|)$ *is fulfilled.*

Remarks I

The proof methods developed so far are not directly applicable to the nondeterministic case. Looking at the proofs of Theorems 7 and 8 in Lecture 7, we see that we have taken advantage of the fact that deterministic complexity classes are closed under complement.

Remarks I

The proof methods developed so far are not directly applicable to the nondeterministic case. Looking at the proofs of Theorems 7 and 8 in Lecture 7, we see that we have taken advantage of the fact that deterministic complexity classes are closed under complement.

More precisely, if $\mathcal{C}$ is a complexity class, then we set

$$\text{co-}\mathcal{C} =_{df} \{L \mid L \subseteq \Sigma^*,\ \Sigma^* \setminus L \in \mathcal{C}\}\,.$$

We say that a complexity class is *closed under complement* if co-$\mathcal{C} \subseteq \mathcal{C}$.

Remarks II

While it was fairly easy to complement deterministic complexity classes by just returning ‘no’ for all accepting computations, and ‘yes’ to all rejecting computations, this method does not work for nondeterministic complexity classes. The fact that a particular computation did not succeed, is no guarantee that others do not, so the strategy above could put some strings both in the language and in its complement.

Remarks II

While it was fairly easy to complement deterministic complexity classes by just returning ‘no’ for all accepting computations, and ‘yes’ to all rejecting computations, this method does not work for nondeterministic complexity classes. The fact that a particular computation did not succeed, is no guarantee that others do not, so the strategy above could put some strings both in the language and in its complement.

Thus, additional work is needed here. For space complexity there is the famous Immerman–Szelepcsényi Theorem stating that $\mathit{NSPACE}(f(n)) = \text{co-}\mathit{NSPACE}(f(n))$ for all S-constructible bounding functions f satisfying $f(n) \geqslant \log n$ for all $n \in \mathbb{N}$ (cf. Theorem 4).

Remarks III

But for nondeterministic time complexity classes no such theorem is known. The best one has is the almost trivial result given by the following exercise:

Exercise 1. *Let* $f\colon \mathbb{N} \to \mathbb{N}$ *be any* T*-constructible bounding function. Then we have*

$$\text{co-}NTIME(f(n)) \subseteq \bigcup_{c>0} TIME\left(2^{cf(n)}\right) \subseteq \bigcup_{c>0} NTIME\left(2^{cf(n)}\right).$$

Nevertheless, we can show a hierarchy theorem for nondeterministic time complexity classes.

A Complexity Hierarchy for Nonderterministic Time I

We continue with the following lemma that will help us to circumvent the problem whether or not nondeterministic time complexity classes are closed under complement:

Lemma 1 (Seiferas, Fischer, Meyer (1978))

For every NTM M_i there is a constant c_i and an NTM M_k such that

$$\begin{aligned} L(M_k) &= \{w \mid w \in \Sigma^*,\ \mathit{bin}(k) * w \in L(M_i)\} \quad \text{and} \\ T_{M_k}(w) &\leqslant c_i \cdot T_{M_i}(\mathit{bin}(k) * w) \quad \text{for all } w \in \Sigma^* . \end{aligned}$$

For a proof, the reader is referred to the book.

Note that the lemma above may be regarded as a variant of the fixed point theorem.

Complexity Hierarchy for Nonderterministic Time II

Using the latter lemma, we can prove the following hierarchy theorem for nondeterministic time complexity classes. Before presenting it, we also need to prove that there are arbitrarily complex languages acceptable by NTMs. This is done via the following exercise:

Exercise 2. *Let* g *be any* T*-constructible bounding function and let* $M = [B, Z, A]$ *be any NTM such that* $T_M(w) \leqslant g(|w|)$*. Then there is a deterministic TM* M' *such that* $L(M) = L(M')$ *and* $T_{M'}(w) \leqslant g(|w|) \cdot k^{g(|w|)}$*, where* $k = 2 \cdot |B|$*.*

Complexity Hierarchy for Nonderterministic Time III

Having the result of Exercise 2 on hand, the determinisitic Hierarchy Theorem directly implies via Exercise 19 that there are arbitrarily complex languages for nondeterministic time and nondeterministic space.

Theorem 3 (Seiferas, Fischer, Meyer (1978))

Let $g\colon \mathbb{N} \to \mathbb{N}$ *be any* T*-constructible bounding function and let* $f\colon \mathbb{N} \to \mathbb{N}$ *be any bounding function such that* $f(n) = o(g(n))$ *and* $f(n+1) = O(g(n))$. *Then we have*

$$\mathit{NTIME}(f(n)) \subset \mathit{NTIME}(g(n)) .$$

For a proof, the reader is referred to the book.

The Immerman–Szelepcsényi Theorem I

We are going to prove a nondeterministic space hierarchy theorem. Fortunately enough, after three decades of failure, in 1988 it could be proved that nondeterministic space is closed under complement. Even more interestingly, in 1988 two proofs have been published independently of one another by Immerman and Szelepcsényi, respectively. So, we continue with the Immerman–Szelepcsényi Theorem.

The Immerman–Szelepcsényi Theorem I

We are going to prove a nondeterministic space hierarchy theorem. Fortunately enough, after three decades of failure, in 1988 it could be proved that nondeterministic space is closed under complement. Even more interestingly, in 1988 two proofs have been published independently of one another by Immerman and Szelepcsényi, respectively. So, we continue with the Immerman–Szelepcsényi Theorem.

Theorem 4 (Immerman–Szelepcsényi)

co-$NSPACE(f(n)) = NSPACE(f(n))$ *for every S-constructible bounding function* f *satisfying* $f(n) \geqslant \log n$ *for all* $n \in \mathbb{N}$.

The Immerman–Szelepcsényi Theorem II

Proof. Let $L \in \mathit{NSPACE}(f(n))$ be witnessed, without loss of generality, by a nondeterministic 2-tape TM $M = [B, Z, A]$ with $S_M(n) \leqslant f(n)$ for all $n \in \mathbb{N}$.
Let C_x denote the set of $f(|x|)$-space bounded configurations of M on input x. Note that

$$|C_x| \leqslant |Z| \cdot \frac{|B|^{f(|x|)+1} - 1}{|B| - 1} \cdot f(|x|) \; .$$

Furthermore, we write $C_x(\tau)$ to denote the set of configurations that can be reached by M on input x within precisely τ steps of computation. For the ease of notation, we write $c_x(\tau)$ to denote the cardinality of $C_x(\tau)$, i.e., $c_x(\tau) = |C_x(\tau)|$.

The Immerman–Szelepcsényi Theorem III

Without loss of generality, we can also assume that there is a uniquely determined accepting configuration C_a and that M works in each of its computations on input x exactly t steps, where $t \leqslant |C_x|$. The proof of this statement is an exercise. Then we have

$$\begin{aligned} M \text{ does not accept } x \quad & \text{iff} \quad C_a \notin C_x(t) \\ & \text{iff} \quad C_x(t) \text{ contains } c_x(t) \text{ many rejecting} \\ & \qquad \text{final configurations.} \end{aligned}$$

The latter observation can be generalized to

$$\begin{aligned} C_a \notin C_x(\tau) \quad \text{iff} \quad & C_x(\tau) \text{ contains} \\ & c_x(\tau) \text{ many configurations } C' \neq C_a \,. \qquad (1) \end{aligned}$$

The Immerman–Szelepcsényi Theorem IV

Note that the time needed to check Condition (1) is at least $c_x(\tau)$ which is clearly exponential in τ.
We have to construct an NTM M' that accepts $\overline{L}$ $(= \Sigma^* \setminus L)$ in space at most $f(n)$.

The Immerman–Szelepcsényi Theorem IV

Note that the time needed to check Condition (1) is at least $c_x(\tau)$ which is clearly exponential in τ.
We have to construct an NTM M' that accepts $\overline{L}$ $(= \Sigma^* \setminus L)$ in space at most $f(n)$.

The idea for the construction of M' is given by the Equivalence (1) displayed above. Having the additional information $c_x(\tau)$ on hand, one can decide nondeterministically in space $f(|x|)$ whether or not $C_a \in C_x(\tau)$.

The Immerman–Szelepcsényi Theorem IV

Note that the time needed to check Condition (1) is at least $c_x(\tau)$ which is clearly exponential in τ.
We have to construct an NTM M' that accepts $\overline{L}$ $(= \Sigma^* \setminus L)$ in space at most $f(n)$.

The idea for the construction of M' is given by the Equivalence (1) displayed above. Having the additional information $c_x(\tau)$ on hand, one can decide nondeterministically in space $f(|x|)$ whether or not $C_a \in C_x(\tau)$.

Let $C_0(x)$ denote the initial configuration of M on input x. If $x \notin \overline{L}$, the positive answer is easily obtained by guessing a sequence of configurations $C_1, \ldots, C_\tau$ such that $C_1 = C_0(x)$ and C_{i+1} is reachable by M in one step from C_i, and $C_\tau = C_a$.

The Immerman–Szelepcsényi Theorem V

Of course, we have to do this guessing iteratively. That is, we always have at most two configurations on the work tape. This can be easily achieved by deleting C_{i-1} as soon as C_i is generated and verified. Again, for the sake of notation, we write $C_i \vdash C_{i+1}$ if C_{i+1} can be reached by M in one step.

The Immerman–Szelepcsényi Theorem V

Of course, we have to do this guessing iteratively. That is, we always have at most two configurations on the work tape. This can be easily achieved by deleting C_{i-1} as soon as C_i is generated and verified. Again, for the sake of notation, we write $C_i \vdash C_{i+1}$ if C_{i+1} can be reached by M in one step.

For the negative answer, we successively check for $C' \in \mathcal{C}_x$ with $C' \neq C_a$ whether $C' \in \mathcal{C}_x(\tau)$. The number of positive answers is counted (that is the reason, why we need $f(n) \geqslant \log n$). If we find $c_x(\tau)$ many such configurations C', we must conclude $C_a \notin \mathcal{C}_x(\tau)$.

The Immerman–Szelepcsényi Theorem V

Of course, we have to do this guessing iteratively. That is, we always have at most two configurations on the work tape. This can be easily achieved by deleting C_{i-1} as soon as C_i is generated and verified. Again, for the sake of notation, we write $C_i \vdash C_{i+1}$ if C_{i+1} can be reached by M in one step.

For the negative answer, we successively check for $C' \in \mathcal{C}_x$ with $C' \neq C_a$ whether $C' \in \mathcal{C}_x(\tau)$. The number of positive answers is counted (that is the reason, why we need $f(n) \geqslant \log n$). If we find $c_x(\tau)$ many such configurations C', we must conclude $C_a \notin \mathcal{C}_x(\tau)$.

All what is left is to formalize this idea appropriately.

The Immerman–Szelepcsényi Theorem VI

We define a procedure REACH. On input $\tau, c, C_1, \ldots, C_\tau$, REACH tries to decide whether at least one of the configurations $C_1, \ldots, C_\tau$ is in $C_x(\tau)$. If REACH does not succeed, it returns "?". The parameter c is used to estimate the cardinality of $C_x(\tau)$.

```
Procedure REACH($\tau, c, C_1, \ldots, C_\tau$)
Input: $\tau$, $c \in \mathbb{N}$, $\{C_1, \ldots, C_\tau\} \subseteq C_x$
Output: true, false or ?
Method: number := 0;
for $C \in C_x$ in lexicographical order do begin
    guess nondeterministically a computation
        $C_0(x) \vdash D_1 \vdash \cdots \vdash D_\tau$ of length $\tau$;
    if $D_\tau = C$ then number := number + 1;
    if $D_\tau \in \{C_1, \ldots, C_\tau\}$ then return REACH($\tau, c, C_1, \ldots, C_\tau$) = true;
    end;
if number = c then return REACH($\tau, c, C_1, \ldots, C_\tau$) = false;
if number < c then return REACH($\tau, c, C_1, \ldots, C_\tau$) = ?;
end REACH
```

The Immerman–Szelepcsényi Theorem VII

Now, we can show the following lemma:

Lemma 2

If procedure `REACH` *is called with* $c = c_x(\tau)$ *and* `REACH` *returns* `true` *or* `false` *then this answer is correct.*

The Immerman–Szelepcsényi Theorem VII

Now, we can show the following lemma:

Lemma 2

If procedure `REACH` *is called with* $c = c_x(\tau)$ *and* `REACH` *returns* `true` *or* `false` *then this answer is correct.*

Proof. First, assume `true` is returned. This can happen if and only if `REACH` has found an initial segment $C_0(x) \vdash D_1 \vdash \cdots \vdash D_\tau$ of a computation such that $D_\tau \in \{C_1, \ldots, C_\tau\}$. So, the answer is correct.
Next, assume `false` is returned. Then we have *number* $= c = c_x(\tau)$, i.e., `REACH` has found c many different configurations none of which is equal to one of the C_is. By definition of $c_x(\tau)$ this implies that all configurations in $C_x(\tau)$ have been tested, thus $C_x(\tau) \cap \{C_1, \ldots, C_\tau\} = \emptyset$. Hence, the answer is correct. ∎

The Immerman–Szelepcsényi Theorem IX

If `REACH` returns ? for $c = c_x(\tau)$, then not all configurations $C \in C_x(\tau)$ have been found in the `for`-loop. Therefore, one can neither conclude $C_i \notin C_x(\tau)$ nor $C_i \in C_x(\tau)$.

The Immerman–Szelepcsényi Theorem IX

If `REACH` returns ? for $c = c_x(\tau)$, then not all configurations $C \in C_x(\tau)$ have been found in the `for`-loop. Therefore, one can neither conclude $C_i \notin C_x(\tau)$ nor $C_i \in C_x(\tau)$.

Since *number* $\leqslant |C_x|$, one can represent *number* in space $O(f(|x|)$. Moreover, every configuration $C \in C_x(\tau)$ needs space at most $f(|x|)$, and thus `REACH` can be realized by an NTM that obeys the space bound $f(n)$.

The Immerman–Szelepcsényi Theorem IX

If `REACH` returns ? for $c = c_x(\tau)$, then not all configurations $C \in C_x(\tau)$ have been found in the `for`-loop. Therefore, one can neither conclude $C_i \notin C_x(\tau)$ nor $C_i \in C_x(\tau)$.

Since *number* $\leqslant |C_x|$, one can represent *number* in space $O(f(|x|)$. Moreover, every configuration $C \in C_x(\tau)$ needs space at most $f(|x|)$, and thus `REACH` can be realized by an NTM that obeys the space bound $f(n)$.

Next, we have to deal with the question how the numbers $c_x(\tau)$ are determined. This is done iteratively by the following procedure `COUNT`:

The Immerman–Szelepcsényi Theorem X

```
Procedure COUNT($\tau, c$)
Input: $\tau$, $c \in \mathbb{N}$
Output: $d \in \mathbb{N}$ or ?
Method: $d := 0$;
for $C \in C_x$ in lexicographical order do begin
    compute the direct predecessors $C_1, \ldots, C_\tau$ of $C$;
    $z :=$ REACH($\tau - 1, c, C_1, \ldots, C_\tau$);
    if $z =$ true then $d := d + 1$;
    if $z =$ ? then return COUNT($\tau, c$) = ?;
    end;
return COUNT($\tau, c$) := $d$;
end COUNT.
```

The Immerman–Szelepcsényi Theorem XI

Lemma 3

If COUNT *is called with* $c = c_x(\tau - 1)$ *and returns a natural number* d*, then* $d = c_x(\tau)$.

The Immerman–Szelepcsényi Theorem XI

Lemma 3

If COUNT *is called with* $c = c_x(\tau - 1)$ *and returns a natural number* d*, then* $d = c_x(\tau)$.

Proof. If COUNT is not stopping with output “?” received from REACH, then, by Lemma 2, REACH correctly answers each question whether one of the predecessors C_i of C belongs to $C_x(\tau - 1)$. Thus, after completion of the for-loop, the value of d coincides with $|C_x(\tau)|$, since every $C \in C_x(\tau)$ must have a predecessor in $C_x(\tau - 1)$. ∎

The Immerman–Szelepcsényi Theorem XI

Lemma 3

If COUNT *is called with* $c = c_x(\tau - 1)$ *and returns a natural number* d, *then* $d = c_x(\tau)$.

Proof. If COUNT is not stopping with output "?" received from REACH, then, by Lemma 2, REACH correctly answers each question whether one of the predecessors C_i of C belongs to $C_x(\tau - 1)$. Thus, after completion of the for-loop, the value of d coincides with $|C_x(\tau)|$, since every $C \in C_x(\tau)$ must have a predecessor in $C_x(\tau - 1)$. ∎

Moreover, it is easy to verify that COUNT can be executed in space $O(f(|x|))$.

The Immerman–Szelepcsényi Theorem XII

Now, we can put this all together to decide nondeterministically whether or not M, on input any x does *not* possess an accepting computation.

```
On input $x$ do the following:
$c_x(0) := 1$;
for $\tau = 1, \ldots, t$ do
    if $c_x(\tau - 1) \neq ?$ then $c_x(\tau) := \mathtt{COUNT}(\tau, c_x(\tau - 1))$;
    else $c_x(\tau) := ?$ ;
if $c_x(\tau) \neq ?$ then $z := \mathtt{REACH}(t, c_x(t), C_a)$;
if $z = \mathtt{false}$ then accept $x$.
```

The Immerman–Szelepcsényi Theorem XIII

This program part can be executed by an NTM M' in space $O(f(|x|))$, too.

The Immerman–Szelepcsényi Theorem XIII

This program part can be executed by an NTM M' in space $O(f(|x|))$, too.

Furthermore, if M' is accepting an input x, then there must be a computation path of M' such that neither `REACH` nor `COUNT` do return "?". Hence, `COUNT` correctly computes the values $c_x(\tau)$ for $\tau = 1, \ldots, t$ and `REACH` verifies that $C_a \notin C_x(t)$. Consequently, $x \notin L$, thus $x \in \overline{L}$ and we are done.

The Immerman–Szelepcsényi Theorem XIV

Finally, if $x \notin L$, then for every configuration $C \in \mathcal{C}_x(\tau)$ there is partial computation $C_0(x) \vdash D_1 \vdash \cdots \vdash D_\tau = C$ that can be guessed while processing

$$\texttt{REACH}(\tau, c_x(\tau), C_1, \ldots, C_\tau)\ .$$

Consequently, `COUNT` is correctly computing the values $c_x(1), \ldots, c_x(t)$, too. But then $\texttt{REACH}(t, c_x(t), C_a)$ must return `false`. Therefore, M' is accepting x.
Thus, we have shown that $\overline{L} \in \text{co-}\mathit{NSPACE}(f(n))$ implies $\overline{L} \in \mathit{NSPACE}(f(n))$, i.e.,

$$\text{co-}\mathit{NSPACE}(f(n)) \subseteq \mathit{NSPACE}(f(n))$$

for all S-constructible bounding functions f satisfying $f(n) \geqslant \log n$ for all $n \in \mathbb{N}$. Hence, we can conclude

$$\text{co-}\mathit{NSPACE}(f(n)) = \mathit{NSPACE}(f(n)) \qquad \text{by Exercise 3 below}\ .$$

Note that the S-constructibility of f ensures that $s = f(|x|)$ can be determined by M' and thus, also t is known. ∎

The Immerman–Szelepcsényi Theorem XV

A closer look at the proof of Theorem 4 shows that an easy modification at the end of the proof even allows for proving the following more general corollary[1]:

Corollary 2

co-$NSPACE^{\max}(f(n)) \subseteq NSPACE^{\max}(f(n))$ for all bounding functions f satisfying $f(n) \geqslant \log n$ for all $n \in \mathbb{N}$.

We leave it as an exercise to show Corollary 2. Furthermore as already mentioned in the proof of Theorem 4, the following result always holds:

Exercise 3. *For every complexity class $\mathcal{C}$ we have, if co-$\mathcal{C} \subseteq \mathcal{C}$ then co-$\mathcal{C} = \mathcal{C}$.*

[1] The complexity class $NSPACE^{\max}(f(n))$ is obtained when "minimum" is everywhere replaced by "maximum" in the Definition of $S(w)$.

Hierarchy for Nondeterministic Space

Now, it is easy to prove the following hierarchy result for nondeterministic space:

Theorem 5

Let $f, g\colon \mathbb{N} \to \mathbb{N}$ be any two bounding functions such that $f(n) = o(g(n))$, $g(n) \geqslant \log n$ for all $n \in \mathbb{N}$ and g is S-constructible. Then

$$\mathit{NSPACE}(f(n)) \subset \mathit{NSPACE}(g(n)) \,.$$

The proof is left as an exercise.

Remark

One final remark is in order here. In all our hierarchy theorems, we have always required the bounding function of the larger complexity class to be constructible. This condition cannot be dropped. For non-constructible functions one can prove nice gap theorems. For giving us a flavor, we finally include the following exercise here:

Exercise 4. *There are recursive functions* $f,\ g\colon \mathbb{N} \to \mathbb{N}$ *such that*

(1) $\mathit{TIME}(f(n)) = \mathit{TIME}(2^{f(n)})$;

(2) $\mathit{SPACE}(g(n)) = \mathit{SPACE}(2^{2^{g(n)}})$.

LBA I

After having seen that nondeterministic space is closed under complement provided that the bounding function f satisfies $f(n) \geqslant \log n$ for all $n \in \mathbb{N}$, we can answer the question whether or not the context-sensitive languages are closed under complement. This is done by characterizing $\mathcal{CS}$ in terms of complexity theory.

LBA I

After having seen that nondeterministic space is closed under complement provided that the bounding function f satisfies $f(n) \geqslant \log n$ for all $n \in \mathbb{N}$, we can answer the question whether or not the context-sensitive languages are closed under complement. This is done by characterizing $\mathcal{CS}$ in terms of complexity theory.

Definition 1

A *linear bounded automaton* is a nondeterministic one-tape TM such that

(1) its input alphabet contains two special symbols ¶ and \$ which are used to mark the leftmost and rightmost position of the tape, respectively, that can be reached by the head.

(2) The head can neither replace ¶ nor \$.

LBA II

So, a linear bounded automaton can only use the space between the two markers ¶ and \$. The input is written between these markers. Please note that the end-markers are written on the tape together with the input but are themselves not considered to belong to the input. Now, taking Theorem 10.11 from *Theory of Computation* into account, we can easily show the following:

Theorem 6

For every context-sensitive language L *there is a linear bounded automaton* M *such that* $L = L(M)$.

LBA III

Proof. First, we divide the tape into two traces. The upper trace contains the input w which will not be changed during the computation. The lower trace is actually used to simulate a possible derivation of w provided it exists. Both traces are uniformly marked by ¶ and \$.

LBA III

Proof. First, we divide the tape into two traces. The upper trace contains the input w which will not be changed during the computation. The lower trace is actually used to simulate a possible derivation of w provided it exists. Both traces are uniformly marked by ¶ and $.

Let $L = L(\mathcal{G})$, where $\mathcal{G} = [T, N, \sigma, P]$ is without loss of generality length increasing. (cf. ToC, Theorem 10.11). If the input is λ, M just stops in its accepting state. If the input is not λ, M starts its computation by writing σ on the leftmost place in the lower trace. Then M guesses nondeterministically a derivation. If the derivation yields the input w, the input string w is accepted. Otherwise, M stops without accepting the input.

LBA IV

To formalize this idea, we have to think about a way how this nondeterministic guessing is performed. This is done by guessing a production and a position on the tape. From the production it can be derived how many cells are needed to replace the nonterminal at the guessed position. If the guessed position does not contain a nonterminal in the lower trace, nothing is done and a new guess is made. If there is a nonterminal at the guessed position, the substring to the right of the position is moved by the number of cells needed to replace the nonterminal. If there is enough space to perform this shift, then the replacement is done. If there is not enough space, then M stops without accepting.

LBA V

Finally, if a string has been derived that is precisely as long as the input, it is compared with the input in the upper trace. If both strings are identical, M accepts. Otherwise, it stops without accepting. Since $\mathcal{G}$ is length increasing, there cannot be a derivation for w that exceeds the space between the markers ¶ and \$. Therefore, M accepts a string w if and only if $w \in L(\mathcal{G})$. ∎

LBA V

Finally, if a string has been derived that is precisely as long as the input, it is compared with the input in the upper trace. If both strings are identical, M accepts. Otherwise, it stops without accepting. Since $\mathcal{G}$ is length increasing, there cannot be a derivation for w that exceeds the space between the markers ¶ and \$. Therefore, M accepts a string w if and only if $w \in L(\mathcal{G})$. ∎

Interestingly enough, the converse direction is also true. That is, every language accepted by a linear bounded automaton is context-sensitive. Thus, we have the following result:

LBA VI

Theorem 7

If $L = L(M)$ *for a linear bounded automaton* M, *then* $L \in \mathcal{CS}$.

The proof is not too hard and left as an exercise.

LBA VI

Theorem 7

If $L = L(M)$ *for a linear bounded automaton* M, *then* $L \in \mathcal{CS}$.

The proof is not too hard and left as an exercise.

Thus we could characterize the context-sensitive languages as the set of all those languages that are accepted by a linear bounded automaton. Since a linear bounded automaton uses space $|w| + 2$ on its tape, we can apply the Immerman–Szelepcsényi Theorem and obtain the affirmative answer to the problem whether or not $\mathcal{CS}$ is closed under complement.

Corollary 3

$\mathcal{CS}$ *is closed under complement.*

Important Complexity Classes I

Now, we are ready to introduce the following important complexity classes:

$$
\begin{aligned}
\mathcal{L} &=_{df} \mathit{SPACE}(\log n)\,; \\
\mathcal{NL} &=_{df} \mathit{NSPACE}(\log n)\,; \\
\mathcal{P} &=_{df} \mathit{TIME}(n^{O(1)}) = \bigcup_{c\in\mathbb{N}} \mathit{TIME}(n^c)\,; \\
\mathcal{NP} &=_{df} \mathit{NTIME}(n^{O(1)}) = \bigcup_{c\in\mathbb{N}} \mathit{NTIME}(n^c)\,; \\
\mathcal{PSPACE} &=_{df} \mathit{SPACE}(n^{O(1)}) = \bigcup_{c\in\mathbb{N}} \mathit{SPACE}(n^c)\,; \\
\mathcal{NPSPACE} &=_{df} \mathit{NSPACE}(n^{O(1)}) = \bigcup_{c\in\mathbb{N}} \mathit{NSPACE}(n^c)\,.
\end{aligned}
$$

Important Complexity Classes II

We can immediately make the following observations:

Theorem 2

(1) $\mathcal{L} \subseteq \mathcal{NL}$;

(2) $\mathcal{P} \subseteq \mathcal{NP}$;

(3) $\mathcal{PSPACE} \subseteq \mathcal{NPSPACE}$;

(4) $\mathcal{P} \subseteq \mathcal{PSPACE}$;

(5) $\mathcal{NP} \subseteq \mathcal{NPSPACE}$.

Important Complexity Classes II

We can immediately make the following observations:

Theorem 2

(1) $\mathcal{L} \subseteq \mathcal{NL}$;

(2) $\mathcal{P} \subseteq \mathcal{NP}$;

(3) $\mathcal{PSPACE} \subseteq \mathcal{NPSPACE}$;

(4) $\mathcal{P} \subseteq \mathcal{PSPACE}$;

(5) $\mathcal{NP} \subseteq \mathcal{NPSPACE}$.

Moreover, we already know the following proper inclusion:

Theorem 8

$\mathcal{L} \subset \mathcal{PSPACE}$.

Important Complexity Classes III

Furthermore, one is often interested in *simultaneously* bounding the time and space resources needed to accept a language. Thus, we shall also study the following complexity classes:

$$\begin{aligned}
\mathit{DTISP}(f(n), g(n)) =_{df} \{L(M) \mid & \text{ M is a DTM } \wedge\ T_M(n) \leqslant f(n) \\
& \wedge\ S_M(n) \leqslant g(n)\}\,; \\
\mathit{NDTISP}(f(n), g(n)) =_{df} \{L(M) \mid & \text{ M is an NDTM} \\
& \wedge\ T_M(n) \leqslant f(n)\ \wedge\ S_M(n) \leqslant g(n)\}\,; \\
\mathcal{PLOPS} =_{df} \mathit{DTISP}\left(n^{O(1)}, (\log n)^{O(1)}\right) & \,.
\end{aligned}$$

Important Complexity Classes IV

Algorithms belonging to the complexity class $\mathcal{PLOPS}$ are considered to be practically realizable using current computer technology. Nevertheless, even this statement has to be read with care, since in practice one has also to ensure that the exponents and constants involved are moderate.

Thank you!

Joel I. Seiferas

Michael J. Fischer

Albert R. Meyer

Neil Immerman