Complexity Theory - Tutorial

Ivan Gavran

December 13th, 2016

1. Prove that L is closed under Kleene star if and only if L = NL.
   Solution We (you) proved in the homework that NL is closed under Kleene star which gives us the direction L = NL $\Rightarrow$ L closed under Kleene star. For the other direction, consider the following NL complete language: ORDERED-PATH = $\{(G = (V, E), s, t)$: there is a path from s to t and $(v_i, v_j) \in E \Rightarrow i < j\}$. The NL-hardness of this language is visible from the proof of NP-hardness of PATH: a configuration graph is acyclic and therefore for each new node we add to the graph the given condition would hold. Let's now define $B := \{Gv_{i+1}Gv_{i+2} \dots Gv_j : (v_i, v_j) \in E(G)\}$. $B \in L$ (we can check whether G repeats checking it bit-by-bit without using any additional space). Furthermore, the string $r = Gv_{s+1}Gv_{s+2} \dots Gv_t \in B^*$ if and only if there is a path from s to t. (Indeed, if there is a path from s to t, it is of the form $v_i, v_{i+k_1}, v_{i+k_2}, \dots, v_j$, because of the ordered condition on graph G. If $r \in B^*$, there is a decomposition of r into

   $$(Gv_{s+1}, \dots Gv_{s+k_1}, Gv_{s+k_2} \dots Gv_{s+k_3}, \dots, Gv_{s+k_m} \dots Gv_{s+k_j})$$

   such that $(v_s, v_{s+k_1}), (v_{s+k_1}, v_{s+k_2}), \dots (v_{s+k_m}, v_j) \in E$.) Therefore, given an instance of ORDERED-PATH, we can transform it to an instance of B^*, which is logspace computable (by the assumption of L being closed on Kleene star). This gives us L=NL.

2. Define the class PP' similarly to PP: $L \in PP'$ if there exists a PPT N such that if $\alpha \in L$, $\mathbb{P}[N(\alpha) = 1] > \frac{1}{2}$, but for $\alpha \notin L$, $\mathbb{P}[N(\alpha) = 0] \geq \frac{1}{2}$ (in other words, PP' accepts by majority, but rejects by non-minority). Show that PP = PP'.
   Solution The direction PP $\subset$ PP$'$ follows from the definitions (a PP machine is at the same time PP' machine). Let $L \in$ PP$'$ and M' a PP' machine for L. Let M' run in time $\geq f(n), \forall(n)$. Now let's design M, a PP machine for L that accepts input α

   toss a coin $f(n) + 1$ times
   if all tosses give tail **then**
   REJECT
   else
   simulate M' on α
   end if

Consider both cases

- if $\alpha \in L$, $\mathbb{P}[M(\alpha) = 1] = \mathbb{P}[M'(\alpha) = 1] \cdot (1 - \frac{1}{2^{f(n)+1}})$. Since M' accepts with probability greater than 0.5, and there are k branches, $k \leq f(n)$, we have the following: $\mathbb{P}[M(\alpha) = 1] \geq (\frac{1}{2} + \frac{1}{2^k}) \cdot (1 - \frac{1}{2^{f(n)+1}}) = \frac{1}{2} - \frac{1}{2^{f(n)+2}} + \frac{1}{2^k} - \frac{1}{2^{f(n)+k+1}} > \frac{1}{2}$
- if $\alpha \notin L$: $\mathbb{P}[M(\alpha) = 0] = \frac{1}{2^{f(n)+1}} + \mathbb{P}[M'(\alpha) = 0] \cdot (1 - \frac{1}{2^{f(n)+1}}) \geq \frac{1}{2^{f(n)+1}} + \frac{1}{2} \cdot (1 - \frac{1}{2^{f(n)+1}}) > \frac{1}{2}$.

Therefore, the constructed machine M is also an PP-machine.

3. Define yet another similar class - PP$''$. $L \in$ PP$''$ if there exists a PPT N such that for $\alpha \in \Sigma^*$, $\mathbb{P}[N(\alpha) = L(\alpha)] \geq \frac{1}{2}$. Show that PP$''$ is not so similar to PP, rather PP$'' = \Sigma^*$
Solution Take any language and let's design machine M for it. On input α, simply toss one coin. If it is head, accept, else reject. $\mathbb{P}[L(\alpha) = M(\alpha] = \frac{1}{2}$. We see that $L \in$ PP$''$

4. For all $k \geq 1$, define class PP$_k$ such that $L \in$ PP$_k$ if there exists a PPT N such that

   - $\alpha \in L \Rightarrow \mathbb{P}[N(\alpha) = 1] > 2^{-k}$
   - $\alpha \notin L \Rightarrow \mathbb{P}[N(\alpha) = 0] \geq 1 - s^{-k}$

   To phrase it differently, N accepts α if and only if more that a fraction of 2^{-k} branches are accepting. Show that PP$_k$ = PP
   Solution Note that PP$_1$ = PP According to previous exercise, It is enough to show PP$_k$ = PP, for $k > 1$. In order to prove equality, we have to prove two inclusions.

   - PP$_k \subset$ PP$'$: Let M_k be a PP$_k$ machine for a language L. We will try to describe a (PP') machine M' on input α.

     make k+1 coin tosses and treat them as binary representation of integer r, $0 \leq r \leq 2^{k+1} - 1$
     if $0 \leq r \leq 2^k - 2$ **then**
       accept
     else if $r = 2^k - 1$ **then**
       reject
     else if $2^k \leq r \leq 2^{k+1} - 1$ **then**
       simulate M_k on α
     end if

For $\alpha \in L$,

$$\begin{aligned}
\mathbb{P}[M'(\alpha) = 1] &= \frac{2^k - 1}{2^{k+1}} + \frac{2^{k+1} - 2^k}{2^{k+1}} \cdot \mathbb{P}[M_k(\alpha) = 1] \\
&= \frac{1}{2} - \frac{1}{2^{k+1}} + \frac{1}{2} \cdot \mathbb{P}[M_k(\alpha) = 1] \\
&> \frac{1}{2} - \frac{1}{2^{k+1}} + \frac{1}{2} \cdot \frac{1}{2^k} \\
&= \frac{1}{2}
\end{aligned}$$

For $\alpha \notin L$,

$$\begin{aligned}
\mathbb{P}[M'(\alpha) = 0] &= \frac{1}{2^{k+1}} + \frac{1}{2} \cdot \mathbb{P}[M_k(\alpha) = 0] \\
&\geq \frac{1}{2^{k+1}} + \frac{1}{2} \cdot (1 - \frac{1}{2^k}) \\
&= \frac{1}{2}
\end{aligned}$$

- $\mathrm{PP}' \subset \mathrm{PP}_k$ Now let M' be a PP' machine for language L, and we'll design a PP_k machine M_k for L (for any given k).

  make k-1 coin tosses
  if all outcomes are heads **then**
  simulate M' on α
  else
  reject
  end if

  Similarly as in the previous case we first observe the case when input $\alpha \in L$

$$\begin{aligned}
\mathbb{P}[M_k(\alpha) = 1] &= \frac{1}{2^{k-1}} \cdot \mathbb{P}[M'(\alpha) = 1] \\
&> \frac{1}{2^{k-1}} \cdot \frac{1}{2}
\end{aligned}$$

  Conversely, if $\alpha \notin L$

$$\begin{aligned}
\mathbb{P}[N_k(\alpha) = 1] &= (1 - \frac{1}{2^{k-1}}) + \frac{1}{2^{k+1}} \cdot \mathbb{P}[M'(\alpha) = 0] \\
&\geq 1 - \frac{1}{2^{k-1}} + \frac{1}{2^k} \\
&= 1 - \frac{1}{2^k}
\end{aligned}$$

5. Prove that PP $\subset$ PSPACE

   Let's start from a language $L \in$ PP and N, a nondeterministic algorithm. We will try to create a deterministic version of the same algorithm that

uses only poly-space. Let $p(n)$ be a polynomial bound on running time of N. This also means that in the run of algorithm there are at most $p(n)$ coin tosses. The algorithm works as follows

```
c_0, c_1 ← 0
for each outcome of p(n) tosses do
  run algorithm N with current state of the tosses
  if N accepts then
    c_0 ← c_0 + 1
  else
    c_1 ← c_1 + 1
  end if
end for
if  c_0 > c_1  then
  accept
else
  reject
end if
```

All runs of N reuse the space. $c_0, c_1 < 2^{p(n)}$, which means that their binary representation is polynomial. Therefore, the whole algorithm runs in polynomial space. Notes
Note Define RP

6. Show that RP is closed under union and intersection
Solution Let M_1 and M_2 be the two RP machines deciding L_1 and L_2.

- intersection: consider the following algorithm ($M_\cap$)

  ```
  Simulate N_1 on input α
  if N_1 rejects then
    reject
  else
    Simulate N_2 on α
    if N_2 accepts then
      accept
    else
      reject
    end if
  end if
  ```

  If $\alpha \in M_1 \cap M_2$, M_cap accepts with probability $\frac{1}{4}$. If $\alpha \notin M_1 \cap M_2$, M_cap it would reject with probability 1.

- union: Consider the following algorithm, M_cup:

  ```
  toss a coin
  if head then
    simulate M_1 on the input α
  else
    simulate M_2 on the input α
  ```

end if

In the case that $\alpha \in L_1 \cup L_2$, the probability of accepting equals $\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}$. Otherwise, α is not in L_1 nor in L_2, therefore there are no accepting branches of either M_1 or M_2 and the probability of accepting is again 0.

NOTE: Sources of the problems and more problems of the same kind:

- `http://blog.computationalcomplexity.org/2015/04/is-logarithmic-space-closed-under.html`
- `http://cse.iitkgp.ac.in/~abhij/course/theory/CC/Spring04/chap5.pdf`