CS 2401 - Introduction to Complexity Theory

Lecture #4: Fall, 2015

Lecturer: Toniann Pitassi

Scribe Notes by: Shuvomoy Das Gupta

1 Review

Recall the definitions of the classes **NP**, and **coNP**.

Definition. *(The class NP)* A language $L \subseteq \{0,1\}^*$ is in **NP**, if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M such that for any $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \exists y \in \{0,1\}^{p(|x|)} \big(M(x,y) = 1\big).$$

For example, `SAT` is in **NP**.

Defintion. *(The class coNP)* A language $L \subseteq \{0,1\}^*$ is in **coNP**, if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$, and a polynomial-time TM M such that for any $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \forall y \in \{0,1\}^{p(|x|)} \big(M(x,y) = 1\big).$$

For example, `TAUTOLOGY` is in **coNP**. Note that in the definition of the class **coNP**, we have $\forall$ as the quantifier rather than $\exists$ as in **NP**.

2 Polynomial hierarchy

Today we are going to talk about polynomial hierarchy. At first we define two new complexity classes: the class Σ_2^P, and the class Π_2^P.

Definition. *(The class Σ_2^P)* The class Σ_2^P is the set of all languages L for which theres exists a polynomial-time TM M, and a polynomial q such that for any $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \exists y_1 \in \{0,1\}^{q(|x|)}\ \forall y_2 \in \{0,1\}^{q(|x|)}\ \left(M(x,y_1,y_2) = 1\right).$$

Definition. *(The class Π_2^P)* The class Σ_2^P is the set of all languages L for which theres exists a polynomial-time TM M, and a polynomial q such that for any $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \forall y_1 \in \{0,1\}^{q(|x|)}\ \exists y_2 \in \{0,1\}^{q(|x|)}\ \left(M(x,y_1,y_2) = 1\right).$$

Note that the difference in the definitions above is the order of the quantifiers; in the class Σ_2^P we have the quantifiers appearing in the order $\exists, \forall$, whereas in the class Π_2^P the order is $\forall, \exists$.

The definition of the polynomial hierarchy generalizes the definitions of **NP**, **coNP**, Σ_2^P and Π_2^P. It consists of every language that can be defined by a combination of a polynomial-time computable

predicate and a fixed number of quantifiers.

Definition. *(The class Σ_i^P)* For $i \geq 1$, a language L is in Σ_i^P if there exists a polynomial-time TM M and a polynomial q such that

$$x \in L \Leftrightarrow \exists y_1 \in \{0,1\}^{q(|x|)}\ \forall y_2 \in \{0,1\}^{q(|x|)}\ \ldots Q_i y_i \in \{0,1\}^{q(|x|)}\ (M(x, y_1, \ldots, y_i) = 1)\,,$$

where Q_i is either $\forall$ (if i is even) or $\exists$ (if i is odd).

Definition. *(The class Π_i^P)* For $i \geq 1$, a language L is in Π_i^P if there exists a polynomial-time TM M and a polynomial q such that

$$x \in L \Leftrightarrow \forall y_1 \in \{0,1\}^{q(|x|)}\ \exists y_2 \in \{0,1\}^{q(|x|)}\ \ldots Q_i y_i \in \{0,1\}^{q(|x|)}\ (M(x, y_1, \ldots, y_i) = 1)\,,$$

where Q_i is either $\forall$ (if i is odd) or $\exists$ (if i is even).

In the class Σ_i^P we have the quantifiers appearing in the order $\exists, \forall, \exists, \forall, \ldots$, whereas in the class Π_i^P the order is $\forall, \exists, \forall, \exists, \ldots$; so the difference is in the order of the quantifiers.

Definition. *(The polynomial hierarchy)* The polynomial hierarchy is the set $\mathbf{PH} = \bigcup_i \Sigma_i^P$.

Note that $\Sigma_1^P = \mathbf{NP}$, and $\Pi_i^P = \mathbf{coNP}$. Also, for every i, we have $\Sigma_i^P \subseteq \Pi_{i+1}^P \subseteq \Sigma_{i+2}^P \subseteq \cdots$. Hence, $\mathbf{PH} = \bigcup_i \Pi_i^P$. So we have, $\mathbf{PH} = \bigcup_i \Sigma_i^P = \bigcup_i \Pi_i^P$, $\mathbf{NP} \subseteq \mathbf{PH}$ and $\mathbf{P} \subseteq \mathbf{PH}$.

Complete problems for PH. A quantified Boolean formula (QBF) is a formula of the form

$$Q_1 x_1\ Q_2 x_2\ \ldots\ Q_n x_n \quad \phi(x_1, x_2, \ldots, x_n),$$

where, each Q_i is one of the two quantifiers $\forall$ or $\exists$, $x_i \in \{0,1\}$ for any $i \in \{1, \ldots, n\}$ and ϕ is an unquantified Boolean formula. We define the language `TQBF` to be the set of all QBFs which are evaluated true.

For any $i \geq 1$, the class Σ_i^P has the following complete problem, which is special case of `TQBF`, with a fixed number of alternations:

$$\Sigma_i\mathtt{SAT} = \exists x_1 \forall x_2 \exists \ldots Q_i x_i \quad \phi(x_1, x_2, \ldots, x_n) = 1,$$

where Q_i is $\forall$ if i is even and $\exists$ else.

Similarly, for any $i \geq 1$, the class Π_i^P has the following complete problem with a fixed number of alternations:

$$\Pi_i\mathtt{SAT} = \forall x_1 \exists x_2 \forall \ldots Q_i x_i \quad \phi(x_1, x_2, \ldots, x_n) = 1,$$

where Q_i is $\exists$ if i is even and $\forall$ else.

3 Space-bounded computation

Consider a Turing machine with the following configuration:

- Tapes:

−Input tape (read only)
−Work tape (read/ write)

• Finite set Q of possible states containing:

$-q_{\text{start}}$: This is the special starting state of a Turing machine, which we describe as follows. The input tape of a Turing machine initially contains the start symbol $\triangleright$, a finite nonblank symbol x, and the blank symbol $\textvisiblespace$ for the rest of its cells. All heads start at the left ends of the input tape and work tape. The machine is in the special starting state, which we denote by q_{start}. The configuration of the Turing machine associated with this state is called the start configuration on input x.

$-q_{\text{halt}}$: This is a special halting state of a Turing machine, which has the property that once the machine is in q_{halt}, the transition function does not allow the machine to further modify the contents of the tape or change the states. So the Turing machine halts once it enters q_{halt}.

$-q_{\text{accept}}$: This state exists only for a nondeterministic Turing machine. Recall that the only difference between a deterministic and nondeterministic Turing machine is that the latter has two transition functions, denoted by δ_0 and δ_1, and the special state q_{accept}. At each step of computation performed, a nondeterministic Turing machine makes an arbitrary choice as to which one of δ_0 and δ_1 to apply. If there exists some sequence of nondeterministic choices for any input x that would cause the machine to reach the state q_{accept}, then the machine outputs 1 for that input. On the other hand, if every sequence of nondeterministic choices causes the machine to reach q_{halt} before reaching q_{accept} for any input, then the machine outputs 0 for that input. In this regard, both q_{accept} and q_{halt} can be considered as halting states for a nondeterministic Turing machine.

Now we define space-bounded computation for a TM, which we apply only to the work tape. As a result we restrict the definition of space-bounded computation to languages (decision problems with answers confined in *yes* or *no*) only, and exclude function problems. Recall that, function problems require an answer more elaborate than that of decision problems, so for them we have to add a third tape to the TM for writing the output.

Definition. *(Space-bounded computation)* Let $S : \mathbb{N} \to \mathbb{N}$ and $L \in \{0,1\}^*$. We have $L \in \mathbf{SPACE}(S(n))$ if there exists a constant c and a TM M deciding L such that at most $c \cdot S(n)$ locations of M's work tapes (excluding the input tape) are visited by M's head during its computation on every input of length n. Similarly, we have $L \in \mathbf{NSPACE}(S(n))$, if there exists an NDTM M deciding L such that, regardless of its nondeterministic choices, it never uses more than $c \cdot S(n)$ nonblank tape locations on inputs of length n.

Some space complexity classes.

• $\mathbf{L} = \mathbf{SPACE}(\log n)$
• $\mathbf{NL} = \mathbf{NSPACE}(\log n)$
• $\mathbf{PSPACE} = \bigcup_{c>0} \mathbf{SPACE}(n^c)$
• $\mathbf{NPSPACE} = \bigcup_{c>0} \mathbf{NSPACE}(n^c)$

Later we will show that $\mathbf{PSPACE} = \mathbf{NPSPACE}$. First we show that $\mathbf{PSPACE} \subseteq \mathbf{EXP}$.

$\mathbf{PSPACE} \subseteq \mathbf{EXP}$. We can bound the time complexity of a TM in terms of its space complexity. Suppose a language $L \in \mathbf{PSPACE}$, then there exists a polynomial S, and a deterministic TM such that M decides L and halts after using at most $p(n)$ tape squares, where x is the input of length

n. The configuration of M consists of its state, position of the head and the contents of the tape. As M halts, it can never enter the same configuration twice, as otherwise it would enter an infinite loop and would never halt. The number of symbols for the tape alphabet is $|\Gamma|$, and number of states is $|Q|$. The position of the read-write head can be in one of $S(n)$ positions. Each cell of the tape contains one symbol from the alphabet, and the contents of such a cell cannot be modified if it is not visited by the read-write head. So there are $|\Gamma|^{S(n)}$ possibilities for the content of the tape. Hence, in total there are $|Q|S(n)|\Gamma|^{S(n)}$ possible configurations for M during its computation on input x. Let us consider a polynomial V such that $V(n) \geq \log|Q| + \log S(n) + S(n)\log|\Gamma|$, then clearly L can be decided by M in time $2^{V(n)}$. We can also represent $V(n) = c \cdot S(n)$, where c is a constant depending on $|\Gamma|, |Q|$ and number of tapes. Recall that $\mathbf{EXP} = \bigcup_k \mathbf{DTIME}(2^{n^k})$. So, $L \in \mathbf{EXP}$. Thus $\mathbf{PSPACE} \subseteq \mathbf{EXP}$.

Space bounded configuration and $s-t$ connectivity problem. There is a strong link between space bounded configuration and directed graph $s-t$ connectivity problem. The $s-t$ connectivity problem is described as follows. Consider a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. Consider two vertices of the graph, denoted by s (source) and t (sink). We want to determine if there is a directed path from $s$to t.

Now we describe the notion of a *configuration graph* for a TM. Consider a TM denoted by M. A *configuration* of a TM M contains the contents of all nonblank entries of M's tapes, along with its state and the position of the head. For every space $S(n)$ TM M and input $x \in \{0,1\}^*$, the configuration graph of M on input x is denoted by $G_{M,x}$, which is a directed graph. The nodes of $G_{M,x}$ are associated with all possible configurations of M. The input of M contains the value of x, and the work tapes can have at most $S(|x|)$ nonblank cells. The edges of $G_{M,x}$ are constructed as follows. Consider two nodes of $G_{M,x}$ associated with the configuration C and C' of M. There exists an edge from C to C', if C' can be reached from C in one step according to M's transition function. If M is deterministic, then the out-degree of the graph is one; if M is nondeterministic, then the out-degree is at most two, as an NDTM has two transition functions. The start configuration of M is unique and depends on the input x. The node of $G_{M,x}$ associated with this unique start configuration of M is denoted by `start-config`(x). We modify M to erase the contents of its work tapes before it halts. So, we can assume that there is a unique single configuration, denoted by C_{accept}, such that M halts and outputs 1 at it. The input x is accepted by M if and only if there exists a directed path from `start-config`(x) to C_{accept} in $G_{M,x}$.

First, note that every vertex in $G_{M,x}$ can be described using $cS(n)$ bits, where c is a positive constant, and $G_{M,x}$ has at most $2^{cS(n)}$ nodes. We can justify the claim by following the same line of logic presented in the justification of $\mathbf{PSPACE} \subseteq \mathbf{EXP}$. Now deciding if two nodes of M are neighbors can expressed as conjunction of many checks. Each check depends on a constant number of bits. Recall that for every Boolean function $f : \{0,1\}^l \to \{0,1\}$, there exists an l-variable CNF formula ϕ of size $l2^l$ such that $\phi(u) = f(u)$ for every $u \in \{0,1\}^l$, where the the size of a CNF formula is defined to be the number of $\wedge/\vee$ symbols it contains. and can be expressed by constant sized CNF formulas. So, each check can be expressed by constant-sized CNF formulas, with the number of variables being proportional to the size fo the workspace of M. So, there exists an $O(S(n))$-size CNF formula $\phi_{M,x}$ such that for every two nodes C, C' of $G_{M,x}$, we have $\phi_{M,x}(C, C') = 1$ if and only if C and C' encode two neighboring configuration in $G_{M,x}$.

Recall that, A language L is in $\mathbf{DTIME}(S(n)) \Leftrightarrow$There is a TM that runs in time $cS(n) : c > 0$ and decides L. So $\mathbf{DTIME}(S(n)) \subseteq \mathbf{SPACE}(S(n)) \subseteq \mathbf{NSPACE}(S(n))$. Now we show that,

NSPACE$(S(n)) \subseteq$ **DTIME**$(2^{O(S(n))})$. We can enumerate over all possible configurations and construct the graph $G_{M,x}$ in time $2^{O(S(n))}$, and thus check if `start-config`(x) is connected to C_{accept} in $G_{M,x}$. We can do the checking by using breadth-first search algorithm to solve $s-t$ connectivity, which is linear in the size of the graph. Thus we have arrived at the following:

For every space constructible $S : \mathbb{N} \to \mathbb{N}$,**DTIME**$(S(n)) \subseteq$ **SPACE**$(S(n)) \subseteq$ **NSPACE**$(S(n)) \subseteq$ **DTIME**$(2^{O(S(n))})$.

So we have: **L** $\subseteq$ **NL** $\subseteq$ **P** $\subseteq$ **NP** $\subseteq$ **PH** $\subseteq$ **PSPACE** $\subseteq$ **EXP**, as shown in Figure 1.

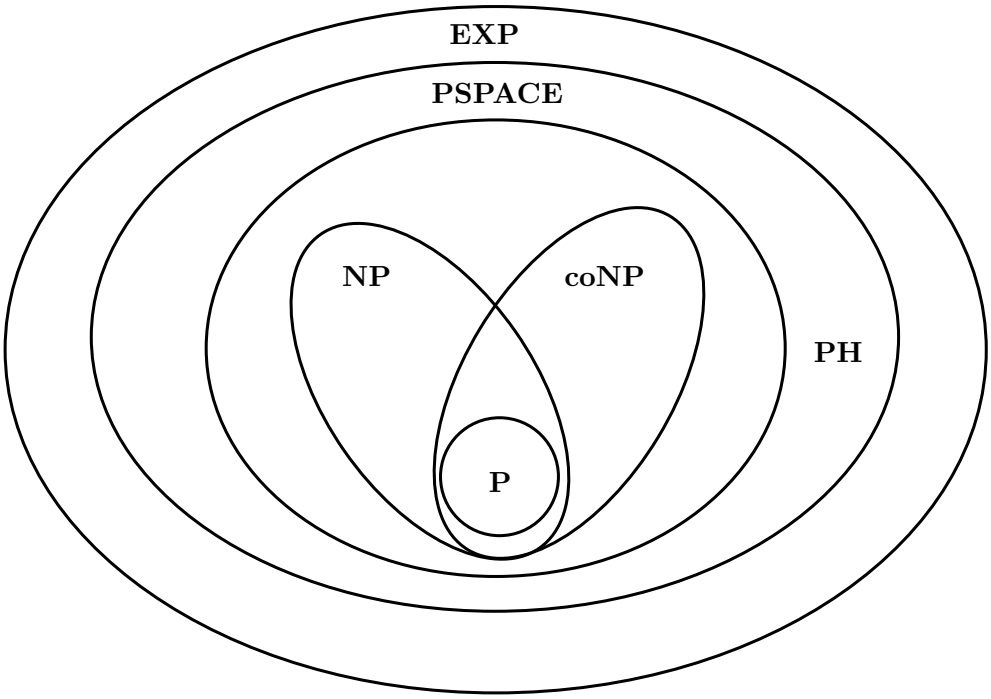

Figure 1: Different complexity classes

4 Complete problems for space classes

We define **PSPACE**-hard and **PSPACE**-complete languages first.

Definition. *(***PSPACE***-hard)* A language L' is **PSPACE**-hard if for every $L \in$ **PSPACE**, $L \leq_p L'$.

Definition. *(***PSPACE***-complete)* A language L' is **PSPACE**-complete if L' is **PSPACE**-hard and $L' \in$ **PSPACE**.

Now we show that `TQBF` is **PSPACE**-space complete. To that goal, we first show that `TQBF` $\in$ **PSPACE**.

Theorem 1. *TQBF is in* **PSPACE**.

Proof. (Arora, pages 84-85) Consider the QBF ψ with n variables

$$\psi = Q_1 x_1 \; Q_2 x_2 \; \ldots \; Q_n x_n \quad \phi(x_1, x_2, \ldots, x_n),$$

where each Q_i is one of the two quantifiers $\forall$ or $\exists$, $x_i \in \{0,1\}$ for any $i \in \{1, \ldots, n\}$ and ϕ is an unquantified Boolean formula. The size of ϕ is m. All possible truth assignments of the variables can be arranged as the leaves of a full binary tree of depth n. Here, the left subtree of the root contains all the truth assignments with $x_1 = 0$. The right subtree of the root contain all the assignments of $x_1 = 1$. Then we branch on x_2, x_3, and so on (see Figure 2).

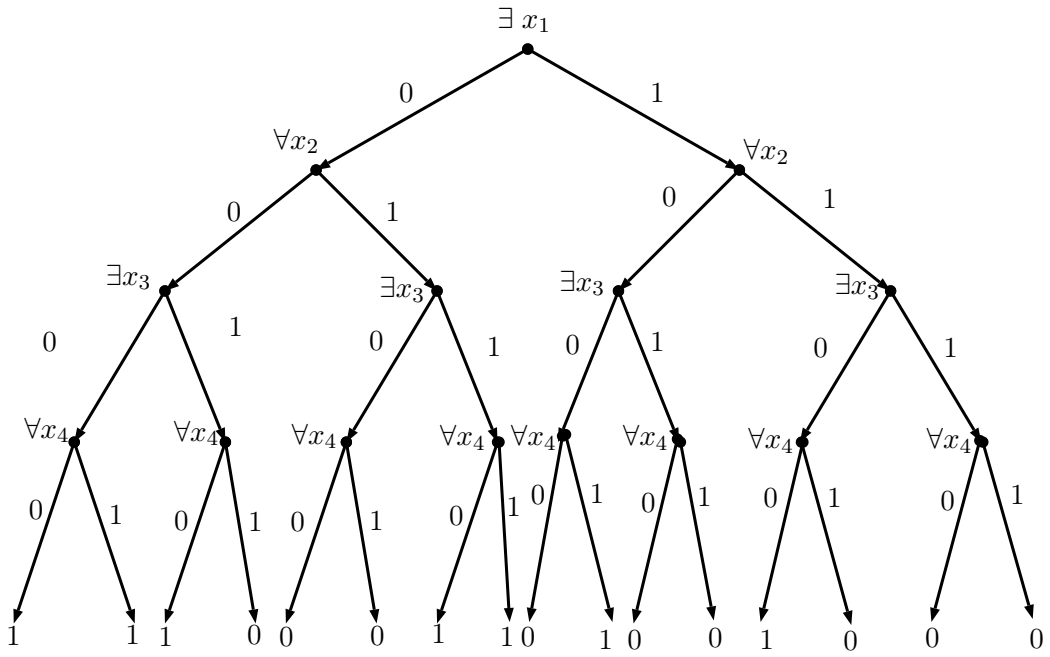

Figure 2: A binary tree for a TQBF with 4 variables

Now we construct a simple recursive algorithm A. Let $s_{n,m}$ be the space used by A. For convenience we assume that the unquantified Boolean formula ϕ contains the variables (of the form x_i), their negation (of the form $\neg x_i$) and constants 0 corresponding to *false*, and 1 corresponding to *true*. If there are no variables ($n = 0$), then the formula contains only the constants. In that case, the formula can be evaluate in $O(m)$ time and space. So, we assume there is at least one variable ($n > 0$). Define

$$\psi_{|x_1=b} = Q_2 x_2 \; \ldots \; Q_n x_n \quad \phi(b, x_2, \ldots, x_n),$$

where $b \in \{0,1\}$. So in $\psi_{|x_1=b}$, we have modified ψ, where the first quantifier Q_1 is dropped and all the occurrences of x_1 in ϕ has been replaced by b. Algorithm A will work as follows.

```
If (Q1 = ∃)
    then (output 1⇔A(ψ|x1=0) = 1 ∨ A(ψ|x1=1))
Else
    (output 1 ⇔ A(ψ|x1=0) = 1 ∧ A(ψ|x1=1))
```

So, A returns the correct answer on any QBF ψ. Note that, space can be reused. After $A(\psi_{|x_1=0})$ is computed, A retains only the single bit of output from the computation, and reuses the space left for the computation of $A(\psi_{|x_1=1})$. So both $A(\psi_{|x_1=0})$ and $A(\psi_{|x_1=1})$ can be run in the same space. If we assume that A uses $O(m)$ space to write $\psi_{|x1=b}$, then $s_{n,m} = s_{n-1,m} + O(m)$. Now we can apply the same line of logic to x_2, x_3, and so on. Thus, $s_{n,m} = O(n \cdot m)$. So, `TQBF` is in **PSPACE**. □