Lecture 12: Polynomial Hierarchy II

Arijit Bishnu

06.04.2010

Outline

Outline

Completeness

Σ_i^p-complete

We say that a language L is Σ_i^p-complete if $L \in \Sigma_i^p$ and for every $L' \in \Sigma_i^p$, $L' \leq_P L$.

Completeness

Σ_i^p-complete

We say that a language L is Σ_i^p-complete if $L \in \Sigma_i^p$ and for every $L' \in \Sigma_i^p$, $L' \leq_P L$.

Examples

Completeness

Σ_i^p-complete

We say that a language L is Σ_i^p-complete if $L \in \Sigma_i^p$ and for every $L' \in \Sigma_i^p$, $L' \leq_P L$.

Examples

- SAT is complete for the class NP $= \Sigma_1^p$.

Completeness

Σ_i^p-complete

We say that a language L is Σ_i^p-complete if $L \in \Sigma_i^p$ and for every $L' \in \Sigma_i^p$, $L' \leq_P L$.

Examples

- SAT is complete for the class NP $= \Sigma_1^p$.
- Recall a QBF has the form $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \ \varphi(x_1, x_2, \ldots, x_n)$ where each $Q_i \in \{\exists, \forall\}$ and φ is an unquantified boolean formula.

Completeness

Σ_i^p-complete

We say that a language L is Σ_i^p-complete if $L \in \Sigma_i^p$ and for every $L' \in \Sigma_i^p$, $L' \leq_P L$.

Examples

- SAT is complete for the class NP $= \Sigma_1^p$.
- Recall a QBF has the form $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \ \varphi(x_1, x_2, \ldots, x_n)$ where each $Q_i \in \{\exists, \forall\}$ and φ is an unquantified boolean formula.
- The language TQBF is the set of QBFs that are TRUE, i.e.

$$\text{TQBF} = \{< \phi > \mid \phi \text{ is a TRUE fully QBF.}\}$$

Examples continued

Examples

Examples continued

Examples

- Define

$$\Sigma_i \text{SAT} = \exists x_1 \forall x_2 \exists \cdots Q_i x_i \ \varphi(x_1, x_2, \ldots, x_i) = 1$$

Examples continued

Examples

- Define

$$\Sigma_i\text{SAT} = \exists x_1 \forall x_2 \exists \cdots Q_i x_i \ \varphi(x_1, x_2, \ldots, x_i) = 1$$

- For every i, Σ_iSAT is a special case of the TQBF problem.

Examples continued

Examples

- Define

$$\Sigma_i\text{SAT} = \exists x_1 \forall x_2 \exists \cdots Q_i x_i \ \varphi(x_1, x_2, \ldots, x_i) = 1$$

- For every i, Σ_iSAT is a special case of the TQBF problem.
- One can prove that for every i, Σ_iSAT is a complete problem for the class Σ_i^p.

Examples continued

Examples

- Define

$$\Sigma_i\text{SAT} = \exists x_1 \forall x_2 \exists \cdots Q_i x_i \ \varphi(x_1, x_2, \ldots, x_i) = 1$$

- For every i, Σ_iSAT is a special case of the TQBF problem.
- One can prove that for every i, Σ_iSAT is a complete problem for the class Σ_i^p.
- Similarly, a problem Π_iSAT can be shown to be Π_i^p-complete.

Examples continued

Examples

- Define

$$\Sigma_i\text{SAT} = \exists x_1 \forall x_2 \exists \cdots Q_i x_i \ \varphi(x_1, x_2, \ldots, x_i) = 1$$

- For every i, Σ_iSAT is a special case of the TQBF problem.
- One can prove that for every i, Σ_iSAT is a complete problem for the class Σ_i^p.
- Similarly, a problem Π_iSAT can be shown to be Π_i^p-complete.
- So, complete problems exist for each class Σ_i^p.

Completeness for PH

Definition: Polynomial Hierarchy

The polynomial hierarchy is the set $\mathrm{PH} = \bigcup_i \Sigma_i^p$.

Completeness for PH

Definition: Polynomial Hierarchy

The polynomial hierarchy is the set $\mathrm{PH} = \bigcup_i \Sigma_i^p$.

PH-complete

We say that a language L is PH-complete if $L \in \mathrm{PH}$ and for every $L' \in \mathrm{PH}$, $L' \leq_P L$.

Completeness for PH

Definition: Polynomial Hierarchy

The polynomial hierarchy is the set $\mathrm{PH} = \bigcup_i \Sigma_i^p$.

PH-complete

We say that a language L is PH-complete if $L \in \mathrm{PH}$ and for every $L' \in \mathrm{PH}$, $L' \leq_P L$.

Does PH have a complete problem?

Each class Σ_i^p has complete problems. What about $\mathrm{PH} = \bigcup_i \Sigma_i^p$?

Completeness for PH

Claim

If there exists a PH-complete language L, then the polynomial hierarchy collapses to some finite level, i.e. $\mathrm{PH} = \Sigma_i^p$.

Completeness for PH

Claim

If there exists a PH-complete language L, then the polynomial hierarchy collapses to some finite level, i.e. $\mathrm{PH} = \Sigma_i^p$.

Proof idea

Completeness for PH

Claim

If there exists a PH-complete language L, then the polynomial hierarchy collapses to some finite level, i.e. $\mathrm{PH} = \Sigma_i^p$.

Proof idea

- Suppose, for some i, $\Sigma_i^p = \Sigma_{i+1}^p$. Then, $\forall j \geq i$, $\Sigma_j^p = \Pi_j^p = \Sigma_i^p$. (Proof as an exercise.)

Completeness for PH

Claim

If there exists a PH-complete language L, then the polynomial hierarchy collapses to some finite level, i.e. $\mathrm{PH} = \Sigma_i^p$.

Proof idea

- Suppose, for some i, $\Sigma_i^p = \Sigma_{i+1}^p$. Then, $\forall j \geq i$, $\Sigma_j^p = \Pi_j^p = \Sigma_i^p$. (Proof as an exercise.)
- Since $L \in \mathrm{PH} = \bigcup_i \Sigma_i^p$, $\exists i$ such that $L \in \Sigma_i^p$.

Completeness for PH

Claim

If there exists a PH-complete language L, then the polynomial hierarchy collapses to some finite level, i.e. $\mathrm{PH} = \Sigma_i^p$.

Proof idea

- Suppose, for some i, $\Sigma_i^p = \Sigma_{i+1}^p$. Then, $\forall j \geq i$, $\Sigma_j^p = \Pi_j^p = \Sigma_i^p$. (Proof as an exercise.)
- Since $L \in \mathrm{PH} = \bigcup_i \Sigma_i^p$, $\exists i$ such that $L \in \Sigma_i^p$.
- Since L is PH-complete, $\forall L' \in \Sigma_{i+1}^p$, $L' \leq_P L$. So, $L' \in \Sigma_i^p$ and $\Sigma_i^p = \Sigma_{i+1}^p$.

Completeness for PH

Claim

If there exists a PH-complete language L, then the polynomial hierarchy collapses to some finite level, i.e. $\text{PH} = \Sigma_i^p$.

Proof idea

- Suppose, for some i, $\Sigma_i^p = \Sigma_{i+1}^p$. Then, $\forall j \geq i$, $\Sigma_j^p = \Pi_j^p = \Sigma_i^p$. (Proof as an exercise.)
- Since $L \in \text{PH} = \bigcup_i \Sigma_i^p$, $\exists i$ such that $L \in \Sigma_i^p$.
- Since L is PH-complete, $\forall L' \in \Sigma_{i+1}^p$, $L' \leq_P L$. So, $L' \in \Sigma_i^p$ and $\Sigma_i^p = \Sigma_{i+1}^p$.
- So, the hierarchy collapses to the level i.

Completeness for PH

Claim

If there exists a PH-complete language L, then the polynomial hierarchy collapses to some finite level, i.e. $\text{PH} = \Sigma_i^p$.

Proof idea

- Suppose, for some i, $\Sigma_i^p = \Sigma_{i+1}^p$. Then, $\forall j \geq i$, $\Sigma_j^p = \Pi_j^p = \Sigma_i^p$. (Proof as an exercise.)
- Since $L \in \text{PH} = \bigcup_i \Sigma_i^p$, $\exists i$ such that $L \in \Sigma_i^p$.
- Since L is PH-complete, $\forall L' \in \Sigma_{i+1}^p$, $L' \leq_P L$. So, $L' \in \Sigma_i^p$ and $\Sigma_i^p = \Sigma_{i+1}^p$.
- So, the hierarchy collapses to the level i.
- So, people believe that complete problems do not exist for the class PH.

Space Complexity Classes and Polynomial Hierarchy

Relation between PH and PSPACE

$PH \subseteq PSPACE$

Space Complexity Classes and Polynomial Hierarchy

Relation between PH and PSPACE

$PH \subseteq PSPACE$

Proof

We can prove just the way we proved $NP \subseteq PSPACE$ by writing down all short (polynomial sized) certificates in the work tape and reusing that space.

Space Complexity Classes and Polynomial Hierarchy

Claim

If PH = PSPACE, the polynomial hierarchy collapses.

Space Complexity Classes and Polynomial Hierarchy

Claim

If PH = PSPACE, the polynomial hierarchy collapses.

Proof

Space Complexity Classes and Polynomial Hierarchy

Claim

If PH = PSPACE, the polynomial hierarchy collapses.

Proof

- We know that TQBF is PSPACE-complete.

Space Complexity Classes and Polynomial Hierarchy

Claim

If PH = PSPACE, the polynomial hierarchy collapses.

Proof

- We know that TQBF is PSPACE-complete.
- Now, if PH = PSPACE, then TQBF is also a complete problem for the class PH.

Space Complexity Classes and Polynomial Hierarchy

Claim

If PH = PSPACE, the polynomial hierarchy collapses.

Proof

- We know that TQBF is PSPACE-complete.
- Now, if PH = PSPACE, then TQBF is also a complete problem for the class PH.
- We have already shown that the existence of a complete problem for the class PH implies that the polynomial hierarchy collapses.

Space Complexity Classes and Polynomial Hierarchy

Claim

If PH = PSPACE, the polynomial hierarchy collapses.

Proof

- We know that TQBF is PSPACE-complete.
- Now, if PH = PSPACE, then TQBF is also a complete problem for the class PH.
- We have already shown that the existence of a complete problem for the class PH implies that the polynomial hierarchy collapses.
- So, the widely held belief is that PH $\neq$ PSPACE.

Outline

Alternating Turing Machines and Classes

Definition: Alternating Time

For every $T : \mathbb{N} \to \mathbb{N}$, we say that an ATM M runs in $T(n)$-time if for every input $x \in \{0,1\}^*$ and for every possible sequence of transition function choices, M halts after at most $T(|x|)$ steps.

Alternating Turing Machines and Classes

Definition: ATIME(T(n))

A language L is in ATIME(T(n)) if there is a constant c and a $c \cdot T(n)$-time ATM M s.t. for every $x \in \{0,1\}^*$, M accepts x iff $x \in L$. The definition of accept comes via labeling vertices on the configuration graph $G_{M,x}$ as follows:

Alternating Turing Machines and Classes

Definition: ATIME(T(n))

A language L is in ATIME(T(n)) if there is a constant c and a $c \cdot T(n)$-time ATM M s.t. for every $x \in \{0,1\}^*$, M accepts x iff $x \in L$. The definition of accept comes via labeling vertices on the configuration graph $G_{M,x}$ as follows:

- The configuration C_{acc} is labeled ACCEPT.

Alternating Turing Machines and Classes

Definition: ATIME(T(n))

A language L is in ATIME(T(n)) if there is a constant c and a $c \cdot T(n)$-time ATM M s.t. for every $x \in \{0,1\}^*$, M accepts x iff $x \in L$. The definition of accept comes via labeling vertices on the configuration graph $G_{M,x}$ as follows:

- The configuration C_{acc} is labeled ACCEPT.
- If a configuration C is in a state that is labeled $\exists$ and there is an edge from C to C' labeled ACCEPT, we label C ACCEPT.

Alternating Turing Machines and Classes

Definition: ATIME(T(n))

A language L is in ATIME(T(n)) if there is a constant c and a $c \cdot T(n)$-time ATM M s.t. for every $x \in \{0,1\}^*$, M accepts x iff $x \in L$. The definition of accept comes via labeling vertices on the configuration graph $G_{M,x}$ as follows:

- The configuration C_{acc} is labeled ACCEPT.
- If a configuration C is in a state that is labeled $\exists$ and there is an edge from C to C' labeled ACCEPT, we label C ACCEPT.
- If a configuration C is in a state labeled $\forall$ and both the configurations C', C'' reachable from it in one step are labeled ACCEPT, then we label C ACCEPT.

Alternating Turing Machines and Classes

Definition: ATIME(T(n))

A language L is in ATIME(T(n)) if there is a constant c and a $c \cdot T(n)$-time ATM M s.t. for every $x \in \{0,1\}^*$, M accepts x iff $x \in L$. The definition of accept comes via labeling vertices on the configuration graph $G_{M,x}$ as follows:

- The configuration C_{acc} is labeled ACCEPT.
- If a configuration C is in a state that is labeled $\exists$ and there is an edge from C to C' labeled ACCEPT, we label C ACCEPT.
- If a configuration C is in a state labeled $\forall$ and both the configurations C', C'' reachable from it in one step are labeled ACCEPT, then we label C ACCEPT.
- We say that M accepts x if this repeated application of labeling rules (till they cannot be applied any more) labels C_s, the start configuration ACCEPT.

ATMs Restricted to a Fixed Number of Alternations

Definition

For every $i \in \mathbb{N}$, we define $\Sigma_i\text{TIME}(T(n))$ ($\Pi_i\text{TIME}(T(n))$) to be the set of languages accepted by a $T(n)$-time ATM M whose initial state is labeled $\exists$ ($\forall$) and for every input and on every directed path from the starting configuration in the configuration graph, M can alternate at most $i-1$ times.

ATMs Restricted to a Fixed Number of Alternations

Definition

For every $i \in \mathbb{N}$, we define $\Sigma_i\text{TIME}(T(n))$ ($\Pi_i\text{TIME}(T(n))$) to be the set of languages accepted by a $T(n)$-time ATM M whose initial state is labeled $\exists$ ($\forall$) and for every input and on every directed path from the starting configuration in the configuration graph, M can alternate at most $i-1$ times.

Classes related to such ATMs

What is $\bigcup_c \Sigma_i\text{TIME}(n^c)$ and $\bigcup_c \Pi_i\text{TIME}(n^c)$?

ATMs Restricted to a Fixed Number of Alternations

Definition

For every $i \in \mathbb{N}$, we define $\Sigma_i\text{TIME}(T(n))$ ($\Pi_i\text{TIME}(T(n))$) to be the set of languages accepted by a $T(n)$-time ATM M whose initial state is labeled $\exists$ ($\forall$) and for every input and on every directed path from the starting configuration in the configuration graph, M can alternate at most $i-1$ times.

Classes related to such ATMs

What is $\bigcup_c \Sigma_i\text{TIME}(n^c)$ and $\bigcup_c \Pi_i\text{TIME}(n^c)$?

Claim

For every $i \in \mathbb{N}$, $\Sigma_i^p = \bigcup_c \Sigma_i\text{TIME}(n^c)$ and $\Pi_i^p = \bigcup_c \Pi_i\text{TIME}(n^c)$.

ATMs Restricted to a Fixed Number of Alternations

Definition

For every $i \in \mathbb{N}$, we define $\Sigma_i\text{TIME}(T(n))$ ($\Pi_i\text{TIME}(T(n))$) to be the set of languages accepted by a $T(n)$-time ATM M whose initial state is labeled $\exists$ ($\forall$) and for every input and on every directed path from the starting configuration in the configuration graph, M can alternate at most $i-1$ times.

Classes related to such ATMs

What is $\bigcup_c \Sigma_i\text{TIME}(n^c)$ and $\bigcup_c \Pi_i\text{TIME}(n^c)$?

Claim

For every $i \in \mathbb{N}$, $\Sigma_i^p = \bigcup_c \Sigma_i\text{TIME}(n^c)$ and $\Pi_i^p = \bigcup_c \Pi_i\text{TIME}(n^c)$.

Proof

Left as an exercise.

Unlimited Number of Alternations

The Class AP

$\mathsf{AP} = \bigcup_c \mathsf{ATIME}(n^c)$.

Unlimited Number of Alternations

The Class AP

$\mathrm{AP} = \bigcup_c \mathrm{ATIME}(n^c)$.

Theorem

$\mathrm{AP} = \mathrm{PSPACE}$

Unlimited Number of Alternations

The Class AP

$\mathrm{AP} = \bigcup_c \mathrm{ATIME}(n^c)$.

Theorem

$\mathrm{AP} = \mathrm{PSPACE}$

Proof Idea

Unlimited Number of Alternations

The Class AP

$AP = \bigcup_c ATIME(n^c)$.

Theorem

$AP = PSPACE$

Proof Idea

- TQBF is a PSPACE-complete problem.

Unlimited Number of Alternations

The Class AP

$AP = \bigcup_c ATIME(n^c)$.

Theorem

$AP = PSPACE$

Proof Idea

- TQBF is a PSPACE-complete problem.
- TQBF $\in$ AP as we can guess using $\exists$ and $\forall$ states of the ATM.

Unlimited Number of Alternations

The Class AP

$\mathrm{AP} = \bigcup_c \mathrm{ATIME}(n^c)$.

Theorem

$\mathrm{AP} = \mathrm{PSPACE}$

Proof Idea

- TQBF is a PSPACE-complete problem.
- TQBF $\in$ AP as we can guess using $\exists$ and $\forall$ states of the ATM.
- So, AP $\subseteq$ PSPACE.

Unlimited Number of Alternations

The Class AP

$AP = \bigcup_c ATIME(n^c)$.

Theorem

$AP = PSPACE$

Proof Idea

- TQBF is a PSPACE-complete problem.
- TQBF $\in$ AP as we can guess using $\exists$ and $\forall$ states of the ATM.
- So, AP $\subseteq$ PSPACE.
- To show PSPACE $\subseteq$ AP, use a recursive procedure similar . to the one we used for showing TQBF $\in$ PSPACE.

Exercises

Exercise

Consider APSPACE to be the languages accepted by ATMs that run using polynomial space. Show that APSPACE = EXP.

Exercises

Exercise

Consider APSPACE to be the languages accepted by ATMs that run using polynomial space. Show that APSPACE = EXP.

Exercise

Consider AL to be the languages accepted by ATMs that run using logarithmic space. Show that AL = P.

Outline

Time-Space TradeOff

- As we do not know much about the P vs NP question, we even cannot rule out a deterministic poly-time algo for SAT.

Time-Space TradeOff

- As we do not know much about the P vs NP question, we even cannot rule out a deterministic poly-time algo for SAT.
- What is that we can do?

Time-Space TradeOff

- As we do not know much about the P vs NP question, we even cannot rule out a deterministic poly-time algo for SAT.
- What is that we can do?

The Class TISP

For every two functions S, $T : \mathbb{N} \to \mathbb{N}$, the class $\mathrm{TISP}(T(n), S(n))$ is the set of languages decided by a TM M that on every input $x \in \{0,1\}^*$ takes at most $O(T(|x|))$ steps and uses at most $O(S(|x|))$ cells of its read/write tape.

Time-Space TradeOff

- As we do not know much about the P vs NP question, we even cannot rule out a deterministic poly-time algo for SAT.
- What is that we can do?

The Class TISP

For every two functions S, $T : \mathbb{N} \to \mathbb{N}$, the class $\mathrm{TISP}(T(n), S(n))$ is the set of languages decided by a TM M that on every input $x \in \{0,1\}^*$ takes at most $O(T(|x|))$ steps and uses at most $O(S(|x|))$ cells of its read/write tape.

SAT and TISP

One can show that $\mathrm{SAT} \notin \mathrm{TISP}(n^c, n^d)$ for every constants c, d such that $c(c+d) < 2$.