**Definition 8.1**

Let $M$ be a deterministic Turing machine, DTM, that halts on all inputs. The space complexity of $M$ is the function $f: \mathbb{N} \rightarrow \mathbb{N}$, where $f(n)$ is the maximum number of tape cells that $M$ scans on any input of length $n$.

If $M$ is a nondeterministic Turing machine, NTM, wherein all branches of its computation halt on all inputs, we define the space complexity of $M$, $f(n)$, to be the maximum number of tape cells that $M$ scans on any branch of its computation for any input of length $n$.

**Example 1 (8.3)**

$SAT$ can be solved with the linear space algorithm $M_1$:

$M_1 =$ “On input $\langle \Phi \rangle$, where $\Phi$ is a Boolean formula:
1. For each truth assignment to the variables $x_1, \ldots, x_n$ of $\Phi$:
   (a) Evaluate $\Phi$ on that truth assignment;
   (b) If $\Phi$ evaluates to 1, accept.
2. If $\Phi$ never evaluates to 1, reject”.

**Estimation of space complexity**

Let $f: \mathbb{N} \rightarrow \mathbb{N}$ be a function. The space complexity classes, $SPACE(f(n))$ and $NSPACE(f(n))$ are defined by:

- $SPACE(f(n)) = \{L \mid L$ is a language decided by an $O(f(n))$ space DTM$\}$
- $NSPACE(f(n)) = \{L \mid L$ is a language decided by an $O(f(n))$ space NTM$\}$
**Construction**

$N =$ “On input $\langle A \rangle$ where $A$ is an NFA:

1. Place a marker on the start state of $A$.
2. Repeat $2^q$ times, where $q$ is the number of states of $A$:
   (a) Nondeterministically select an input symbol and change the position of the markers on $A$’s states to simulate reading of that symbol.
3. If all the marked states are non-final, accept; otherwise reject”.

**Example 2 (8.4)**

Testing whether an NFA accepts all strings,

$\text{ALL}_{\text{NFA}} = \{ \langle A \rangle \mid A \text{ is an NFA and } L(A) = \Sigma^* \}$

$\text{ALL}_{\text{NFA}} = \{ \langle A \rangle \mid A \text{ is an NFA and } L(A) \neq \Sigma^* \}$

We show that $\text{ALL}_{\text{NFA}} \in \text{NSPACE}(n)$.

Proof idea: construct $N$, a nondeterministic linear space algorithm that decide the complement $\overline{\text{ALL}_{\text{NFA}}}$ by guessing a string that is rejected by NFA $A$ and uses a linear space to keep track of which states the NFA could be in at a particular time. Note, this language is not known to be $NP$ or co–$NP$.

**SPACE vs NSPACE**

Recall $\text{NTIME}(f(n)) \subseteq \text{TIME}(2^{O(f(n))})$.

Savitch’s Theorem For any function $f: \mathbb{N} \rightarrow \mathbb{N}$, where $f(n) \geq n$, $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$.

Proof idea: If NTM $N$ uses $f(n)$ space, we need to construct a DTM $M$ which simulates $N$ and uses $f^2(n)$ space.

$M$ calls the function $\text{CANYIELD}(c_1, c_2, t)$, which can be computed by a TM, that tests whether a NTM $N$ can go from the configuration $c_1$ of $N$ to the configuration $c_2$ of $N$ in $t$ steps.
DTM $M$ simulating NTM $N$

Suppose $N$ is modified that once it enters an accepting state, it erases everything on the tape and moves the tape head to the leftmost position. Let such a final configuration be $c_a$.

Let $d$ be a constant such that $N$ has no more than $2^{df(n)}$ configuration.

Then the DTM $M$ is:

$$M = \text{"On input } w$$
1. Let $c_s = q_0w$, where $q_0$ is the start state of $N$.
2. Output the result of $CANYIELD(c_s, c_a, 2^{df(n)})$.

The class $PSPACE$

$PSPACE$ is the class of languages that are decidable in polynomial space on a DTM, i.e.:

$$PSPACE = \bigcup_k \text{SPACE}(n^k)$$

$$NPSPACE = \bigcup_k \text{NSPACE}(n^k)$$

Let us define $EXPTIME = \bigcup_k \text{TIME}(2^{n^k})$

Lemma: $\text{SPACE}(f(n)) \subseteq \text{TIME}(2^{O(f(n))})$.

Analyzing $M$

Whenever $CANYIELD$ invokes itself recursively it stores the current stage number and the values $c_1, c_2, t$ on the stack.

Each level of recursion uses $O(f(n))$ additional space.

Each level of recursion divides the size $t$ in half. Since initially $t = 2^{df(n)}$ the depth of the recursion is $O(\log(2^{df(n)}) = O(f(n))$.

Hence, the total space used is $O(f^2(n))$. 

$CANYIELD(c_1, c_2, t)$

$CANYIELD(c_1, c_2, t)$ is a TM that tests whether a NTM $N$ can go from the configuration $c_1$ of $N$ to the configuration $c_2$ of $N$ in $t$ steps.

$CANYIELD = \text{"On input } c_1, c_2, t:$
1. If $t = 1$ test whether $c_1 = c_2$ or whether $c_1$ yields $c_2$ in one step according to the transition rules of $N$; accept if either test succeeds, reject is both fail.
2. If $t > 1$ then for each configuration $x$ of $N$ using space $f(n)$:
   (a) Run $CANYIELD(c_1, x, t/2)$.
   (b) Run $CANYIELD(x, c_2, t/2)$.
   (c) If both 2(a) and 2(b) accept then accept.
3. If haven’t yet accepted at 2(c), reject.”
The class \textit{PSPACE}

\textit{PSPACE} is the class of languages that are decidable in polynomial space on a DTM, i.e.: 

\[ \text{PSPACE} = \bigcup_k \text{SPACE}(n^k) \]

\[ \text{NPSPACE} = \bigcup_k \text{NSPACE}(n^k) \]

Let us all define \textit{EXPTIME} = \bigcup_k \text{TIME}(2^{O(n^k)})

**Lemma:** \( \text{SPACE}(f(n)) \subseteq \text{TIME}(2^{O(f(n))}) \)

**SUMMARY** \( P \subseteq NP \subseteq \text{PSPACE} = \text{NPSPACE} \subseteq \text{EXPTIME} \)

---

\textbf{PSPACE-Completeness}

A language \( B \) is 'PSPACE-complete' if it satisfies two conditions:

1. \( B \in \text{PSPACE} \)
2. Every \( A \in \text{PSPACE} \) is polynomially time reducible to \( B \).

If \( B \) satisfies only 2, \( B \) is said to be 'PSPACE-hard.'

---

\textbf{TQBF \in PSPACE}

The following polynomial space algorithm decides \textit{TQBF}:

\begin{align*}
T &= "\text{On input } \langle \Phi \rangle, \text{ a fully quantified Boolean formula:} \\
1. &\text{If } \Phi \text{ contains no quantifiers, then it is an expression with only} \\
   &\text{constants. So, evaluate } \Phi \text{ and accept if it is true; otherwise reject.} \\
2. &\text{If } \Phi = \exists x \Psi, \text{ recursively call } T \text{ on } \Psi, \text{ first with 0 substituted for } x \text{ and} \\
   &\text{then with 1 substituted for } x. \text{ If either result is "accept", accept; otherwise reject.} \\
3. &\text{If } \Phi = \forall x \Psi, \text{ recursively call } T \text{ on } \Psi, \text{ first with 0 substituted for } x \text{ and} \\
   &\text{then with 1 substituted for } x. \text{ If both results are "accept", accept; otherwise reject."} \\
\end{align*}

\textit{TQBF} \in \text{PSPACE}

\textbf{A \textit{PSPACE}-complete language}

\textit{TQBF} = \{ \langle \Phi \rangle \mid \Phi \text{ is a true fully quantified Boolean formula} \}

- \( \Phi \) is a fully quantified Boolean formula if
  \( \Phi = Q_1 x_1 Q_2 \cdots Q_n x_n \Psi \), where \( \Psi \) is a Boolean formula in
  \textit{CNF}, \( x_1, x_2, \ldots, x_n \) are the boolean variables in \( \Psi \) (\( \Phi \) is
  said to be in \textit{prenex normal form}), and \( Q_i \in \{ \forall, \exists \} \) for
  \( 1 \leq i \leq n \).

\textit{TQBF} is \textit{PSPACE}-complete.
**Proof Review: SAT is NP-Hard**

- On input $(A, w)$, let $n = |w|$ and $A$ can be recognized by a NTM using $n^k$ time for constant $k$, $f$ will produce $\Phi_w$ in $O(n^k)$.
- **Variables of $\Phi_w$:** Let $N = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, $L(N) = A$, and $C = Q \cup \Gamma \cup \{\#\}$. For each $1 \leq i, j \leq n^k$ and $s \in C$ we have a boolean variable $x_{i,j,s}$ in $\Phi_w$.
- **Cells:** each of the $(n^k)^2$ entries of a tableau is called a cell. $\forall s \in C$ $x_{i,j,s} = 1$ if $\text{cell}[i,j] = s$.
- **Formula $\Phi_w$:** $\Phi_w = \Phi_{\text{cell}} \land \Phi_{\text{start}} \land \Phi_{\text{move}} \land \Phi_{\text{accept}}$.

---

**TQBF is PSPACE-Hard**

Idea: Use $T$ and the construction of $\text{CANYIELD}$ to prove that any $\text{PSPACE}$ problem reduces to $\text{TQBF}$ in polynomial time.

---

**An accepting tableau**

is specified by $\Phi_{\text{start}}, \Phi_{\text{move}}, \Phi_{\text{accept}}$

- $\Phi_{\text{start}}$ ensures that the first row of the tableau is the starting configuration of $N$ on $w$ by the equality:
  $\Phi_{\text{start}} = x_{1,1,\#} \land x_{1,2,q_0} \land x_{1,3,w_1} \land \ldots \land x_{1,n+3,w_n} \land \ldots \land x_{1,n^k-1,w} \land x_{1,n^k,\#}$.
- $\Phi_{\text{accept}}$ guarantees that an accepting configuration occurs in the tableau by placing $q_a$ in one of the cells by: $\Phi_{\text{accept}} = \lor_{1 \leq i,j \leq n^k} x_{i,j,q_a}$.
- $\Phi_{\text{move}}$ guarantees that each row of the tableau correspond to a configuration that legally follows the preceding row's configuration according the $N$'s transition rules.

---

**Assignment-tableau correspondence**

The assignment turns on exactly one variable for each cell, using the following constructs:

1. at least one variable that is associated with a cell is on, by: $\lor_{s \in C} x_{i,j,s}$
2. no more than one variable is on for each each cell, by:
   $\land_{s, t \in C, s \neq t} (x_{i,j,s} \lor x_{i,j,t})$

Thus, any satisfying assignment specifies one symbol in each cell by: $\Phi_{\text{cell}} = \land_{1 \leq i,j \leq n^k} [(\lor_{s \in C} x_{i,j,s}) \land (\land_{s \neq t \in C} (x_{i,j,s} \lor x_{i,j,t}))]$
**TQBF is PSPACE-hard**

Let $A$ be a language decided by TM $M$ in space $n^k$ for some constant $k$. We give a polynomial time reduction from $A$ to TQBF:

The reduction maps machine $M$ and string $w$ to a quantified Boolean formula $\Phi$ that is true iff $M$ accepts $w$:

- The formula is $\Phi_{c_1,c_2,t}$ where $c_1$ and $c_2$ are variables representing two configurations, and $t > 0$.
- If we assign to $c_1$ and $c_2$ actual configurations, $\Phi_{c_1,c_2,t}$ is true iff $M$ can go from $c_1$ to $c_2$ in at most $t$ steps.
- Consider $\Phi_{c_{\text{start}},c_{\text{accept}},h}$, where $h = 2^{d(n)}$ for a constant $d$ chosen so that $M$ has no more than $2^{d(n)}$ configurations. Assume that $t$ is a power of 2.

**Legal windows**

A $2 \times 3$ window of cells is legal if that window does not violate the actions specified by $N$’s transition function. To explain, consider the transitions: $\delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}$.

Examples of legal windows for this machine are:

- **(a)** $\begin{array}{c} x \\ \# \end{array}$ $\begin{array}{c} v_1 \\ q_2 \end{array}$ $\begin{array}{c} b \\ x \end{array}$

- **(b)** $\begin{array}{c} x \\ \# \end{array}$ $\begin{array}{c} v_1 \\ q_2 \end{array}$ $\begin{array}{c} c \\ x \end{array}$

- **(c)** $\begin{array}{c} y \\ \# \end{array}$ $\begin{array}{c} v_1 \\ q_2 \end{array}$ $\begin{array}{c} b \\ z \end{array}$

- **(d)** $\begin{array}{c} y \\ \# \end{array}$ $\begin{array}{c} q_2 \\ v_1 \end{array}$ $\begin{array}{c} c \\ x \end{array}$

- **(e)** $\begin{array}{c} z \\ \# \end{array}$ $\begin{array}{c} q_2 \\ v_1 \end{array}$ $\begin{array}{c} z \\ x \end{array}$

- **(f)** $\begin{array}{c} q_2 \\ \# \end{array}$ $\begin{array}{c} q_2 \\ v_1 \end{array}$ $\begin{array}{c} z \\ a \end{array}$

**A Formula Game**

A formal game is a pair $(X, \Psi)$, where $X$ is a list of boolean variables, $X = [x_1, x_2, \ldots, x_m]$, and $\Psi$ is a quantifier-free boolean formula built on $X$.

The game consists of $m$ moves: In move $i$, a player assigns a boolean value to variable $x_i$.

Two players, $E$ and $A$, take turn to make a move.

If $\Psi$ becomes true after $m$ moves, $E$ wins; otherwise, $A$ wins.

A player is said to have a winning strategy if the player can always win the game by making right moves, no matter how the other player moves.

**Note**

- $\Phi_{c_1,c_2,t}$ encodes the contents of tape cells as in the Cook-Levin theorem. Each configuration has $n^k$ cells and so it is encoded by $O(n^k)$ variables.

- For $t = 1$ the formula $\Phi_{c_1,c_2,t}$ says that either $c_1 = c_2$ or $c_2$ follows from $c_1$ in a single step of $M$.

- If $t > 1$, construct $\Phi_{c_1,c_2,t}$ recursively:
  
  $\Phi_{c_1,c_2,t} = \exists m_1[\Phi_{c_1,m_1,t/2} \land \Phi_{m_1,c_2,t/2}]$ where $m_1$ is a configuration of $M$. $\exists m_1$ is shorthand for $\exists x_1, x_2, \ldots, x_t, t = O(n^k)$ and $x_1, \ldots, x_t$ are variables encoding $m_1$.

- To reduce the size of the formula we use both quantifiers, $\forall$ and $\exists$:

  $\Phi_{c_1,c_2,t} = \exists m_1 \forall(c_3, c_4) \in \{(c_1, m_1), (m_1, c_2)\}[\Phi_{c_3,c_4,t/2}]$ where $\forall x \in \{y, z, \ldots\} \Phi(x)$ denotes $(\Phi(y) \land P(z) \land \ldots)$. 


TQBF vs Formula Game

**Theorem**: Given a game \((X, \Psi)\), suppose \(E\) moves first and \(m\) is even. Let \(\Phi = \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_m[\Psi]\). Then \(\Phi\) is true iff \(E\) has a winning strategy for the game \((X, \Psi)\).

**Lemma**: For any fully quantified boolean formula \(\Phi\), there exists an equivalent fully quantified boolean formula \(\Phi'\) in prenex normal form whose quantifiers alternate between \(\exists\) and \(\forall\).

**Theorem**: For any fully quantified boolean formula \(\Phi\), there exists a formula game \((X, \Psi)\) such that player \(E\) has a winning strategy iff \(\Phi\) is true.

Example of Formula Game

Suppose \(E\) moves first.

- If \(\Phi = (x_1 \lor \bar{x}_2) \land (x_2 \lor x_3) \land (x_2 \lor \bar{x}_3)\) then \(E\) always wins if it selects \(x_1 = 1\), thus \(E\) has the winning strategy.

- If \(\Psi = (x_1 \lor x_2) \land (x_2 \lor x_3) \land (x_2 \lor \bar{x}_3)\) then \(A\) always wins: If \(E\) selects \(x_2 = 1\), then \(A\) selects \(x_1 = 0\); otherwise \(A\) selects \(x_2 = 0\). Thus \(A\) has a winning strategy for this game.

Generalized Geography

This is a child game where players take turns naming cities from anywhere in the world. Each city chosen must begin with the same letter that ended the previous city name and no duplication is allowed.

**Graph model**: A directed graph \(G = (V, E)\) whose nodes are cities of the world and an arrow goes from node \(n_1\) to node \(n_2\) if the city labeling \(n_2\) starts with the letter that ends the name labeling node \(n_1\).

Theorem 8.11

The problem of determining which player has a winning strategy in a formula-game associated with a particular formula is \(PSPACE\)-complete.

**Formally**:

\[\text{FORMULA-GAME} = \{(X, \Phi) | \text{Player E has a winning strategy in the formula game } (X, \Phi)\}\] is \(PSPACE\)-complete.
**The GG problem**

The following algorithm $M$ decides whether player $E$ has a winning strategy for the GG game:

1. If $b$ has outdegree 0, reject, player $E$ loses immediately
2. Remove node $b$ and all connected arrows to get $G'$.
3. For each node $b_1, b_2, \ldots, b_k$ that $b$ originally pointed, recursively call $M$ on $\langle G', b_i \rangle$.
4. If one of $(b_i, G')$ returns "reject", player $E$ would choose $b_i$, and $M$ accepts.
5. If all of these $(b_i, G')$ return "accept", player $A$ has a winning strategy in the original game, so $M$ returns "reject".

**GG is in PSPACE**

Theorem 8.14 $GG$ is PSPACE-complete

Theorem: Formula-Game $\leq_p GG$.

- Let $(X, \Phi)$ be a formula game, where $\Phi$ is in CNF containing $m$ clauses. We construct a graph $G = (V, E)$ and a special node $b$ such that player $E$ has a winning strategy in $(X, \Phi)$ for formula game iff player $E$ has a winning strategy in $(G, b)$ in GG.
- $V = V_1 \cup V_2$, where
  - $V_1 = \{b_i, x_i, \bar{x}_i, c_i \mid 1 \leq i \leq k, k = |X|\}$ (assume $|X|$ is odd), and
  - $V_2 = \{c, c_j, c_{j,i} \mid 1 \leq j \leq m, 1 \leq i \leq k, c_j = (c_{j,1} \lor \cdots \lor c_{j,k})\}$.
- $E = E_1 \cup E_2 \cup E_3$, where
  - $E_1$ constructs a chain of “diamonds” among $V_1$,
  - $E_2$ constructs a tree among $V_2$,
  - $E_3$ connects $V_1$ and $V_2$.
- $b = b_1$

**GG is PSPACE-hard**