# Lecture 9: Space Complexity II

Arijit Bishnu

23.03.2010

# Outline

# Outline

# Savitch's Theorem

- Though a proof is lacking, our intuition says that NP $\neq$ P.

# Savitch's Theorem

- Though a proof is lacking, our intuition says that NP $\neq$ P.
- But, surprisingly it turns out that the same does not carry over to PSPACE and NSPACE. Savitch's Theorem basically states that.

# Savitch's Theorem

- Though a proof is lacking, our intuition says that NP $\neq$ P.
- But, surprisingly it turns out that the same does not carry over to PSPACE and NSPACE. Savitch's Theorem basically states that.
- We have stated already (we will prove it shortly) that TQBF is PSPACE-complete. We can also show that TQBF is NSPACE-complete. Thus, TQBF is in the class of the hardest problem of both classes PSPACE and NSPACE. We already know PSPACE $\subseteq$ NSPACE.

# Savitch's Theorem

- Though a proof is lacking, our intuition says that NP $\neq$ P.
- But, surprisingly it turns out that the same does not carry over to PSPACE and NSPACE. Savitch's Theorem basically states that.
- We have stated already (we will prove it shortly) that TQBF is PSPACE-complete. We can also show that TQBF is NSPACE-complete. Thus, TQBF is in the class of the hardest problem of both classes PSPACE and NSPACE. We already know PSPACE $\subseteq$ NSPACE.
- The above observations suggest PSPACE = NSPACE. Savitch's Theorem formalizes this notion.

# Savitch's Theorem

- Though a proof is lacking, our intuition says that NP $\neq$ P.
- But, surprisingly it turns out that the same does not carry over to PSPACE and NSPACE. Savitch's Theorem basically states that.
- We have stated already (we will prove it shortly) that TQBF is PSPACE-complete. We can also show that TQBF is NSPACE-complete. Thus, TQBF is in the class of the hardest problem of both classes PSPACE and NSPACE. We already know PSPACE $\subseteq$ NSPACE.
- The above observations suggest PSPACE = NSPACE. Savitch's Theorem formalizes this notion.

**Savitch's Theorem**

For any space constructible $S : \mathbb{N} \to \mathbb{N}$ with $S(n) \geq \log n$, $\text{NSPACE}(S(n)) \subseteq \text{SPACE}(S(n)^2)$.

# Recapitulation of the Configuration Graph

## Claim about $G_{M,x}$

Let $G_{M,x}$ be the configuration graph of a space-$S(n)$ machine $M$ on some input $x$ of length $n$. Then,

# Recapitulation of the Configuration Graph

## Claim about $G_{M,x}$

Let $G_{M,x}$ be the configuration graph of a space-$S(n)$ machine $M$ on some input $x$ of length $n$. Then,

- Every vertex in $G_{M,x}$ can be described using $c \cdot S(n)$ bits where $c$ is a constant depending on $M$. $G_{M,x}$ has at most $2^{cS(n)}$ nodes.

# Recapitulation of the Configuration Graph

## Claim about $G_{M,x}$

Let $G_{M,x}$ be the configuration graph of a space-$S(n)$ machine $M$ on some input $x$ of length $n$. Then,

- Every vertex in $G_{M,x}$ can be described using $c \cdot S(n)$ bits where $c$ is a constant depending on $M$. $G_{M,x}$ has at most $2^{cS(n)}$ nodes.
- There is an $O(S(n))$-size CNF formula $\varphi_{M,x}$ such that for every two strings $C$ and $C'$, $\varphi(M,x)(C,C') = 1$ if and only if $C$, $C'$ encode two neighboring configurations in $G_{M,x}$

# Proof Idea for Savitch's Theorem

- Let $L \in \text{NSPACE}(S(n))$. That means an NDTM $N$ decides $L$ using $O(S(n))$ space.

# Proof Idea for Savitch's Theorem

- Let $L \in \text{NSPACE}(S(n))$. That means an NDTM $N$ decides $L$ using $O(S(n))$ space.
- We need to simulate the actions of $N$ by a DTM $M$ that uses $O(S(n)^2)$ space.

# Proof Idea for Savitch's Theorem

- Let $L \in \text{NSPACE}(S(n))$. That means an NDTM $N$ decides $L$ using $O(S(n))$ space.
- We need to simulate the actions of $N$ by a DTM $M$ that uses $O(S(n)^2)$ space.
- Simply simulating the action of all the branches of $N$ might take exponential space.

# Proof Idea for Savitch's Theorem

- Let $L \in \text{NSPACE}(S(n))$. That means an NDTM $N$ decides $L$ using $O(S(n))$ space.
- We need to simulate the actions of $N$ by a DTM $M$ that uses $O(S(n)^2)$ space.
- Simply simulating the action of all the branches of $N$ might take exponential space.
- The trick is to define a new problem that is of recursive nature which is used by $M$ to simulate $N$.

# Proof Idea for Savitch's Theorem

- Let $L \in \text{NSPACE}(S(n))$. That means an NDTM $N$ decides $L$ using $O(S(n))$ space.
- We need to simulate the actions of $N$ by a DTM $M$ that uses $O(S(n)^2)$ space.
- Simply simulating the action of all the branches of $N$ might take exponential space.
- The trick is to define a new problem that is of recursive nature which is used by $M$ to simulate $N$.
- Let $G_{N,x}$ be the configuration graph of $N$ on $x$. The number of vertices in $G_{N,x}$ is $2^{O(S(n))}$ as $N$ uses $O(S(n))$ space.

# Proof Idea for Savitch's Theorem

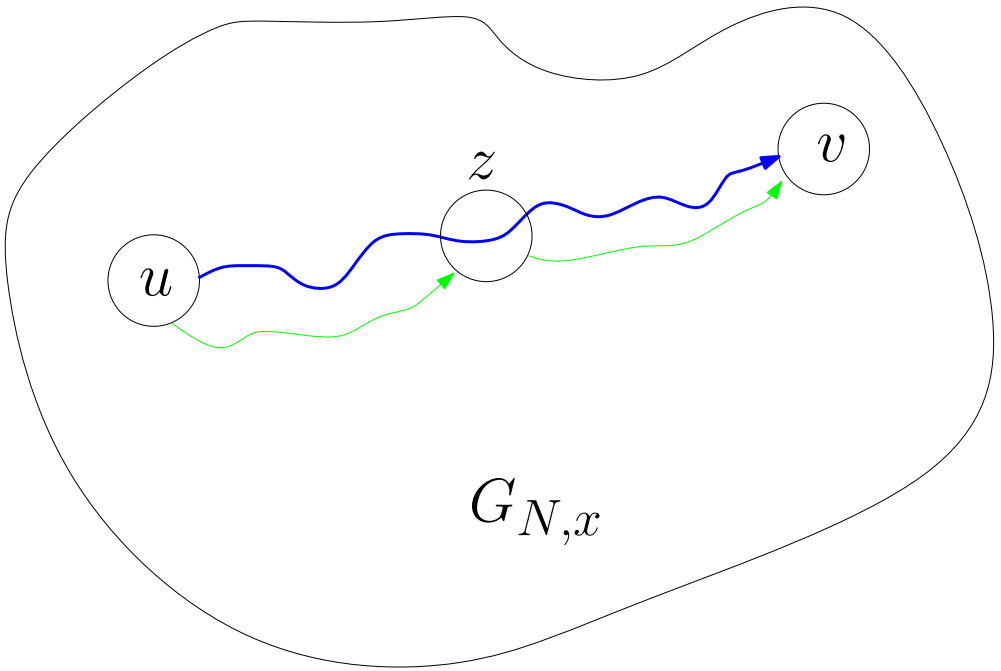
- Let $L \in \text{NSPACE}(S(n))$. That means an NDTM $N$ decides $L$ using $O(S(n))$ space.
- We need to simulate the actions of $N$ by a DTM $M$ that uses $O(S(n)^2)$ space.
- Simply simulating the action of all the branches of $N$ might take exponential space.
- The trick is to define a new problem that is of recursive nature which is used by $M$ to simulate $N$.
- Let $G_{N,x}$ be the configuration graph of $N$ on $x$. The number of vertices in $G_{N,x}$ is $2^{O(S(n))}$ as $N$ uses $O(S(n))$ space.
- The recursive procedure is $\text{REACH}(u, v, i)$ that returns YES if there is a path from $u$ to $v$ of length at most $2^i$ and NO otherwise.

# Proof Idea for Savitch's Theorem

- Let $L \in \text{NSPACE}(S(n))$. That means an NDTM $N$ decides $L$ using $O(S(n))$ space.
- We need to simulate the actions of $N$ by a DTM $M$ that uses $O(S(n)^2)$ space.
- Simply simulating the action of all the branches of $N$ might take exponential space.
- The trick is to define a new problem that is of recursive nature which is used by $M$ to simulate $N$.
- Let $G_{N,x}$ be the configuration graph of $N$ on $x$. The number of vertices in $G_{N,x}$ is $2^{O(S(n))}$ as $N$ uses $O(S(n))$ space.
- The recursive procedure is $\text{REACH}(u, v, i)$ that returns YES if there is a path from $u$ to $v$ of length at most $2^i$ and NO otherwise.
- $\text{REACH}(u, v, \log 2^{O(S(n))})$ where $u = C_s^x$, $v = C_{acc}$, returns YES iff $N$ goes from the start to the accepting state using $O(S(n))$ space.

# Proof Idea continued ...

$$|u \rightsquigarrow v| \leq 2^i$$

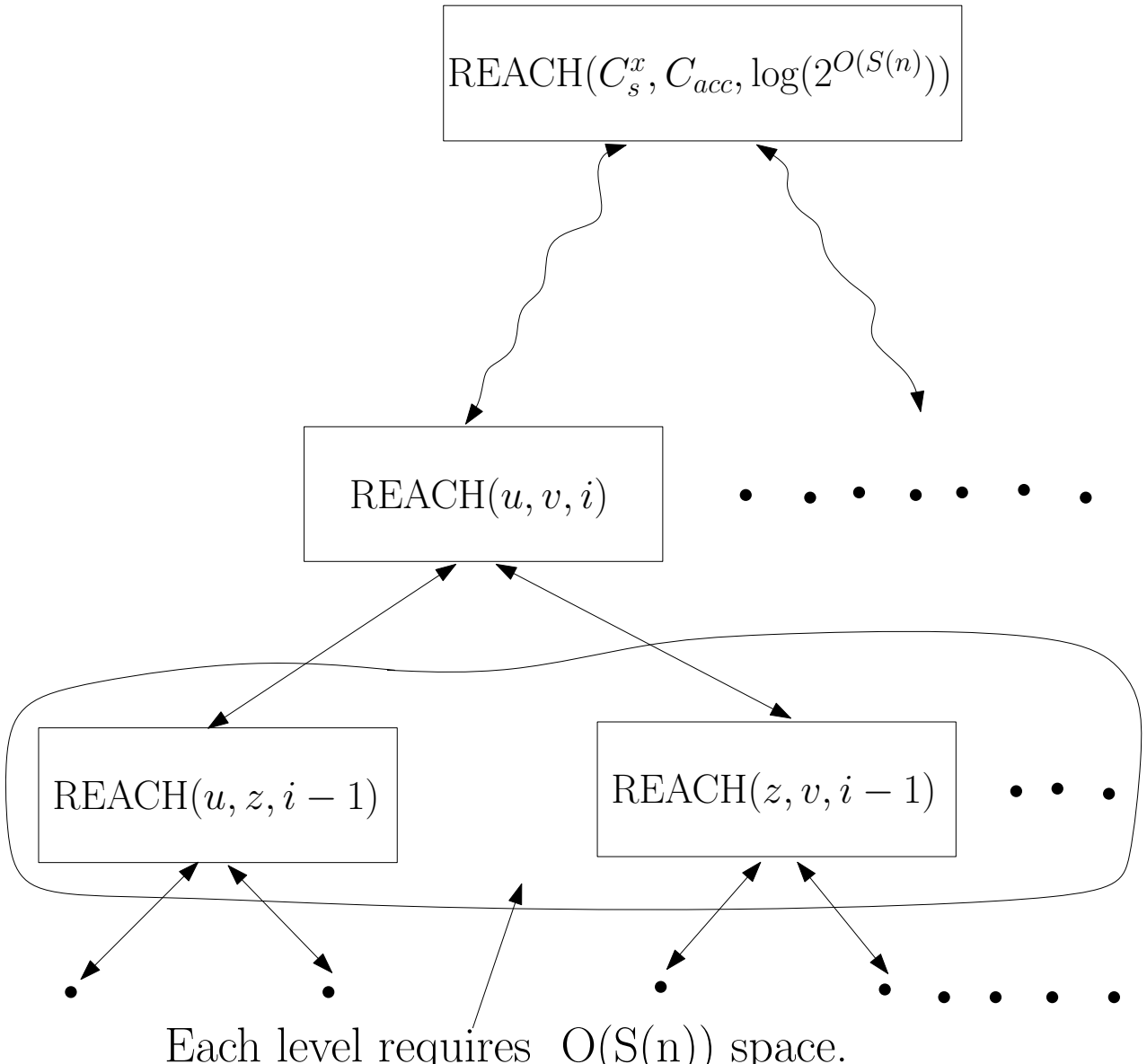



$$\exists z \text{ such that } |u \rightsquigarrow z| = |z \rightsquigarrow v| \leq 2^{i-1}$$

**Observation**

There is a path from $u$ to $v$ of length at most $2^i$ iff $\exists$ a vertex $z$ with paths from $u$ to $z$ and from $z$ to $v$ of lengths at most $2^{i-1}$.

# The Proof


REACH($C_s^x, C_{acc}, \log(2^{O(S(n))})$)
REACH($u, v, i$)
REACH($u, z, i-1$)
REACH($z, v, i-1$)
Each level requires O(S(n)) space.

# The Proof

- On inputs $u, v, i$, the function REACH will enumerate over all vertices $z$ ($2^{O(S(n))}$ in number where each vertex takes $O(S(n))$ bit) in $G_{N,x}$ using $O(\log(2^{O(S(n))})) = O(S(n))$ space and output YES if it finds one $z$ such that REACH$(u, z, i-1)$ = YES and REACH$(z, v, i-1)$ = YES.

# The Proof

- On inputs $u, v, i$, the function REACH will enumerate over all vertices $z$ ($2^{O(S(n))}$ in number where each vertex takes $O(S(n))$ bit) in $G_{N,x}$ using $O(\log(2^{O(S(n))})) = O(S(n))$ space and output YES if it finds one $z$ such that REACH$(u, z, i-1)$ = YES and REACH$(z, v, i-1)$ = YES.
- Think of this computation in terms of the recursion tree.

# The Proof

- On inputs $u, v, i$, the function REACH will enumerate over all vertices $z$ ($2^{O(S(n))}$ in number where each vertex takes $O(S(n))$ bit) in $G_{N,x}$ using $O(\log(2^{O(S(n))})) = O(S(n))$ space and output YES if it finds one $z$ such that REACH$(u, z, i-1)$ = YES and REACH$(z, v, i-1)$ = YES.
- Think of this computation in terms of the recursion tree.
- The depth of the recursion is $O(\log(2^{O(S(n))})) = O(S(n))$.

# The Proof

- On inputs $u, v, i$, the function REACH will enumerate over all vertices $z$ ($2^{O(S(n))}$ in number where each vertex takes $O(S(n))$ bit) in $G_{N,x}$ using $O(\log(2^{O(S(n))})) = O(S(n))$ space and output YES if it finds one $z$ such that REACH$(u, z, i-1)$ = YES and REACH$(z, v, i-1)$ = YES.
- Think of this computation in terms of the recursion tree.
- The depth of the recursion is $O(\log(2^{O(S(n))})) = O(S(n))$.
- As space can be reused, each level of the recursion needs $O(S(n))$ space. (Why?)

# The Proof

- On inputs $u, v, i$, the function REACH will enumerate over all vertices $z$ ($2^{O(S(n))}$ in number where each vertex takes $O(S(n))$ bit) in $G_{N,x}$ using $O(\log(2^{O(S(n))})) = O(S(n))$ space and output YES if it finds one $z$ such that REACH$(u, z, i-1)$ = YES and REACH$(z, v, i-1)$ = YES.
- Think of this computation in terms of the recursion tree.
- The depth of the recursion is $O(\log(2^{O(S(n))})) = O(S(n))$.
- As space can be reused, each level of the recursion needs $O(S(n))$ space. (Why?)
- With $O(S(n))$ levels, the space requirement is $O(S(n)^2)$.

# The Proof

- On inputs $u, v, i$, the function REACH will enumerate over all vertices $z$ ($2^{O(S(n))}$ in number where each vertex takes $O(S(n))$ bit) in $G_{N,x}$ using $O(\log(2^{O(S(n))})) = O(S(n))$ space and output YES if it finds one $z$ such that REACH$(u, z, i-1)$ = YES and REACH$(z, v, i-1)$ = YES.
- Think of this computation in terms of the recursion tree.
- The depth of the recursion is $O(\log(2^{O(S(n))})) = O(S(n))$.
- As space can be reused, each level of the recursion needs $O(S(n))$ space. (Why?)
- With $O(S(n))$ levels, the space requirement is $O(S(n)^2)$.
- Since, $C_{acc}$ is reachable from $C_s^x$ iff it can be reached via a path of length at most $2^{O(S(n))}$, we have the proof.

# Outline

# Quantified Boolean Formula

- A Quantified Boolean Formula (QBF) is a boolean formula in which variables are quantified using $\exists$ and $\forall$.

# Quantified Boolean Formula

- A Quantified Boolean Formula (QBF) is a boolean formula in which variables are quantified using $\exists$ and $\forall$.
- We also specify the universe over which the quantifiers work; in our case it is $\{0, 1\}$.

# Quantified Boolean Formula

- A Quantified Boolean Formula (QBF) is a boolean formula in which variables are quantified using $\exists$ and $\forall$.
- We also specify the universe over which the quantifiers work; in our case it is $\{0, 1\}$.
- Thus, 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 $\varphi$ is an unquantified boolean formula.

# Quantified Boolean Formula

- A Quantified Boolean Formula (QBF) is a boolean formula in which variables are quantified using $\exists$ and $\forall$.
- We also specify the universe over which the quantifiers work; in our case it is $\{0, 1\}$.
- Thus, 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 $\varphi$ is an unquantified boolean formula.
- If there are no free variables, then we say that the QBF is a fully Quantified Boolean Formula. Such a formula is either TRUE or FALSE; there is nothing in between.

# A New Language

## Definition: A New Language TQBF

The language TQBF is the set of QBFs that are TRUE, i.e.

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

# A New Language

**Definition: A New Language TQBF**

The language TQBF is the set of QBFs that are TRUE, i.e.

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

**Theorem**

TQBF is PSPACE-complete.

# A New Language

**Definition: A New Language TQBF**

The language TQBF is the set of QBFs that are TRUE, i.e.

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

**Theorem**

TQBF is PSPACE-complete.

**Proof of TQBF $\in$ PSPACE**

We have already done it.

# TQBF is PSPACE-hard

## Proof of TQBF $\in$ PSPACE-hard

# TQBF is PSPACE-hard

## Proof of TQBF $\in$ PSPACE-hard

- We need to show that every language $L \in$ PSPACE reduces to TQBF in polynomial time.

# TQBF is PSPACE-hard

## Proof of TQBF $\in$ PSPACE-hard

- We need to show that every language $L \in$ PSPACE reduces to TQBF in polynomial time.
- We start with a polynomial space bounded TM $M$ for $L$ and give a poly-time reduction that maps a string to a QBF $\psi$ that encodes the simulation of $M$ on that input.

# TQBF is PSPACE-hard

## Proof of TQBF $\in$ PSPACE-hard

- We need to show that every language $L \in$ PSPACE reduces to TQBF in polynomial time.
- We start with a polynomial space bounded TM $M$ for $L$ and give a poly-time reduction that maps a string to a QBF $\psi$ that encodes the simulation of $M$ on that input.
- The formula is TRUE iff the machine accepts.

# Why doesn't the Cook-Levin idea work?

- Recall the tableau based proof. The tableau was of size $O(n^k \times n^k)$, where the width of $O(n^k)$ represents a configuration and the height represents the number of configurations.

# Why doesn't the Cook-Levin idea work?

- Recall the tableau based proof. The tableau was of size $O(n^k \times n^k)$, where the width of $O(n^k)$ represents a configuration and the height represents the number of configurations.
- Here space can be reused and the machine can run for exponential (exponential in $O(n^k)$) time.

# Why doesn't the Cook-Levin idea work?

- Recall the tableau based proof. The tableau was of size $O(n^k \times n^k)$, where the width of $O(n^k)$ represents a configuration and the height represents the number of configurations.
- Here space can be reused and the machine can run for exponential (exponential in $O(n^k)$) time.
- So, the height of the tableau can be exponential thus giving a boolean formula that will have an exponential nature. This cannot work for a polynomial reduction.

# Why doesn't the Cook-Levin idea work?

- Recall the tableau based proof. The tableau was of size $O(n^k \times n^k)$, where the width of $O(n^k)$ represents a configuration and the height represents the number of configurations.
- Here space can be reused and the machine can run for exponential (exponential in $O(n^k)$) time.
- So, the height of the tableau can be exponential thus giving a boolean formula that will have an exponential nature. This cannot work for a polynomial reduction.
- The idea is to use the same technique as Savitch's theorem.

# A Few Preliminaries

- Assume that $\psi_i(C, C') = \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C')$.

# A Few Preliminaries

- Assume that $\psi_i(C, C') = \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C')$.
- We now introduce two new variables (representing the configurations) $D_1$ and $D_2$ that allows us to fold the two recursive subformulas into a single subformula that preserves the original meaning.

$$\begin{aligned} \psi_i(C, C') &= \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C') \\ &= \exists C'' \forall (D_1, D_2) \in \{(C, C''), (C'', C')\}[\psi_{i-1}(D_1, D_2)] \end{aligned}$$

# A Few Preliminaries

- Assume that $\psi_i(C, C') = \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C')$.
- We now introduce two new variables (representing the configurations) $D_1$ and $D_2$ that allows us to fold the two recursive subformulas into a single subformula that preserves the original meaning.

$$\begin{aligned} \psi_i(C, C') &= \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C') \\ &= \exists C'' \forall (D_1, D_2) \in \{(C, C''), (C'', C')\}[\psi_{i-1}(D_1, D_2)] \end{aligned}$$

# A Few Preliminaries

- Assume that $\psi_i(C, C') = \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C')$.
- We now introduce two new variables (representing the configurations) $D_1$ and $D_2$ that allows us to fold the two recursive subformulas into a single subformula that preserves the original meaning.

$$\begin{aligned} \psi_i(C, C') &= \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C') \\ &= \exists C'' \forall (D_1, D_2) \in \{(C, C''), (C'', C')\} [\psi_{i-1}(D_1, D_2)] \end{aligned}$$

- By writing $\forall (D_1, D_2) \in \{(C, C''), (C'', C')\}$, we indicate that the variables $D_1$ and $D_2$ may take the values of the variables $C$ and $C''$, or $C''$ and $C'$ resp. and the resulting formula $[\psi_{i-1}(D_1, D_2)]$ is TRUE in either case.

# A Few Preliminaries

- Assume that $\psi_i(C, C') = \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C')$.
- We now introduce two new variables (representing the configurations) $D_1$ and $D_2$ that allows us to fold the two recursive subformulas into a single subformula that preserves the original meaning.

$$\begin{aligned} \psi_i(C, C') &= \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C') \\ &= \exists C'' \forall (D_1, D_2) \in \{(C, C''), (C'', C')\} [\psi_{i-1}(D_1, D_2)] \end{aligned}$$

- By writing $\forall (D_1, D_2) \in \{(C, C''), (C'', C')\}$, we indicate that the variables $D_1$ and $D_2$ may take the values of the variables $C$ and $C''$, or $C''$ and $C'$ resp. and the resulting formula $[\psi_{i-1}(D_1, D_2)]$ is TRUE in either case.
- Replace $\forall x \in \{y, z\}[\ldots]$ as $\forall x[((x = y) \vee (x = z)) \Rightarrow \ldots]$. We know $=$ and $\Rightarrow$ can be replaced by $\vee$, $\wedge$ and $\sim$.

# A Few Preliminaries

So, we have

$$\begin{aligned}
\psi_i(C, C') &= \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C') \\
&= \exists C'' \forall (D_1, D_2) \in \{(C, C''), (C'', C')\}[\psi_{i-1}(D_1, D_2)] \\
&= \exists C'' \forall D_1 \forall D_2((\underbrace{D_1 = C} \wedge \underbrace{D_2 = C''}) \vee \\
&\quad (\underbrace{D_1 = C''} \wedge \underbrace{D_2 = C'})) \Rightarrow \psi_{i-1}(D_1, D_2)
\end{aligned}$$

# The Proof: TQBF is PSPACE-hard

- We show that $L \leq_P$ TQBF for every $L \in$ PSPACE. Let $M$ decide $L$ in $S(n)$ space and let $x \in \{0,1\}^n$. We show how to construct a QBF of size $O(S(n)^2)$ that is TRUE iff $M$ accepts $x$.

# The Proof: TQBF is PSPACE-hard

- We show that $L \leq_P$ TQBF for every $L \in$ PSPACE. Let $M$ decide $L$ in $S(n)$ space and let $x \in \{0,1\}^n$. We show how to construct a QBF of size $O(S(n)^2)$ that is TRUE iff $M$ accepts $x$.
- Let $m = O(S(n))$ denote the number of bits needed to encode a configuration of $M$ on an input of length $n$.

# The Proof: TQBF is PSPACE-hard

- We show that $L \leq_P$ TQBF for every $L \in$ PSPACE. Let $M$ decide $L$ in $S(n)$ space and let $x \in \{0,1\}^n$. We show how to construct a QBF of size $O(S(n)^2)$ that is TRUE iff $M$ accepts $x$.
- Let $m = O(S(n))$ denote the number of bits needed to encode a configuration of $M$ on an input of length $n$.
- By the earlier claim on configuration graphs, there is a boolean formula $\varphi_{M,x}$ s.t. for every two strings $C, C' \in \{0,1\}^m$, $\varphi_{M,x}(C, C') = 1$ iff $C$ and $C'$ encode two adjacent configurations in $G_{M,x}$.

# The Proof: TQBF is PSPACE-hard

- We show that $L \leq_P$ TQBF for every $L \in$ PSPACE. Let $M$ decide $L$ in $S(n)$ space and let $x \in \{0,1\}^n$. We show how to construct a QBF of size $O(S(n)^2)$ that is TRUE iff $M$ accepts $x$.
- Let $m = O(S(n))$ denote the number of bits needed to encode a configuration of $M$ on an input of length $n$.
- By the earlier claim on configuration graphs, there is a boolean formula $\varphi_{M,x}$ s.t. for every two strings $C, C' \in \{0,1\}^m$, $\varphi_{M,x}(C, C') = 1$ iff $C$ and $C'$ encode two adjacent configurations in $G_{M,x}$.
- We will use $\varphi_{M,x}$ to come up with a poly-sized QBF $\psi$ that has polynomial number of variables bound by quantifiers and two unquantified variables s.t. for every $C, C' \in \{0,1\}^m$, $\psi(C, C') =$ TRUE iff $\exists$ a directed path from $C$ to $C'$ in $G_{M,x}$.

# The Proof: TQBF is PSPACE-hard

- We show that $L \leq_P$ TQBF for every $L \in$ PSPACE. Let $M$ decide $L$ in $S(n)$ space and let $x \in \{0,1\}^n$. We show how to construct a QBF of size $O(S(n)^2)$ that is TRUE iff $M$ accepts $x$.
- Let $m = O(S(n))$ denote the number of bits needed to encode a configuration of $M$ on an input of length $n$.
- By the earlier claim on configuration graphs, there is a boolean formula $\varphi_{M,x}$ s.t. for every two strings $C, C' \in \{0,1\}^m$, $\varphi_{M,x}(C, C') = 1$ iff $C$ and $C'$ encode two adjacent configurations in $G_{M,x}$.
- We will use $\varphi_{M,x}$ to come up with a poly-sized QBF $\psi$ that has polynomial number of variables bound by quantifiers and two unquantified variables s.t. for every $C, C' \in \{0,1\}^m$, $\psi(C, C') =$ TRUE iff $\exists$ a directed path from $C$ to $C'$ in $G_{M,x}$.
- We plug in the values $C_s^x$ and $C_{acc}$ to $\psi$ to get a QBF that is TRUE iff $M$ accepts $x$.

# The Proof continued: TQBF is PSPACE-hard

- We define $\psi$ inductively. $\psi_0 = \varphi_{M,x}$ and $\psi = \psi_m$. .

# The Proof continued: TQBF is PSPACE-hard

- We define $\psi$ inductively. $\psi_0 = \varphi_{M,x}$ and $\psi = \psi_m$. .
- We let $\psi_i(C, C')$ be TRUE iff $\exists$ a path $C \rightsquigarrow C'$ s.t. $|C \rightsquigarrow C'| \leq 2^i$ in $G_{M,x}$.

# The Proof continued: TQBF is PSPACE-hard

- We define $\psi$ inductively. $\psi_0 = \varphi_{M,x}$ and $\psi = \psi_m$. .
- We let $\psi_i(C, C')$ be TRUE iff $\exists$ a path $C \rightsquigarrow C'$ s.t. $|C \rightsquigarrow C'| \leq 2^i$ in $G_{M,x}$.

**Observation from the proof of Savitch's Theorem**

$\exists$ a path $C \rightsquigarrow C'$ s.t. $|C \rightsquigarrow C'| \leq 2^i$ in $G_{M,x}$ iff $\exists$ a configuration $C''$ s.t. $\exists$ paths $C \rightsquigarrow C''$ and $C'' \rightsquigarrow C'$ both of length at most $2^{i-1}$.

# The Proof continued: TQBF is PSPACE-hard

- We define $\psi$ inductively. $\psi_0 = \varphi_{M,x}$ and $\psi = \psi_m$. .
- We let $\psi_i(C, C')$ be TRUE iff $\exists$ a path $C \rightsquigarrow C'$ s.t. $|C \rightsquigarrow C'| \leq 2^i$ in $G_{M,x}$.

**Observation from the proof of Savitch's Theorem**

$\exists$ a path $C \rightsquigarrow C'$ s.t. $|C \rightsquigarrow C'| \leq 2^i$ in $G_{M,x}$ iff $\exists$ a configuration $C''$ s.t. $\exists$ paths $C \rightsquigarrow C''$ and $C'' \rightsquigarrow C'$ both of length at most $2^{i-1}$.

- The above observation suggests $\psi_i(C, C') = \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C')$.

# The Proof continued: TQBF is PSPACE-hard

- We define $\psi$ inductively. $\psi_0 = \varphi_{M,x}$ and $\psi = \psi_m$. .
- We let $\psi_i(C, C')$ be TRUE iff $\exists$ a path $C \rightsquigarrow C'$ s.t. $|C \rightsquigarrow C'| \leq 2^i$ in $G_{M,x}$.

**Observation from the proof of Savitch's Theorem**

$\exists$ a path $C \rightsquigarrow C'$ s.t. $|C \rightsquigarrow C'| \leq 2^i$ in $G_{M,x}$ iff $\exists$ a configuration $C''$ s.t. $\exists$ paths $C \rightsquigarrow C''$ and $C'' \rightsquigarrow C'$ both of length at most $2^{i-1}$.

- The above observation suggests $\psi_i(C, C') = \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C')$.
- $|\psi_i| = 2\,|\psi_{i-1}|$, and so by induction, one can show that $|\psi_m| = 2^m$ which is exponential and is of no use.

# The Proof continued: TQBF is PSPACE-hard

- We define $\psi$ inductively. $\psi_0 = \varphi_{M,x}$ and $\psi = \psi_m$. .
- We let $\psi_i(C, C')$ be TRUE iff $\exists$ a path $C \rightsquigarrow C'$ s.t. $|C \rightsquigarrow C'| \leq 2^i$ in $G_{M,x}$.

**Observation from the proof of Savitch's Theorem**

$\exists$ a path $C \rightsquigarrow C'$ s.t. $|C \rightsquigarrow C'| \leq 2^i$ in $G_{M,x}$ iff $\exists$ a configuration $C''$ s.t. $\exists$ paths $C \rightsquigarrow C''$ and $C'' \rightsquigarrow C'$ both of length at most $2^{i-1}$.

- The above observation suggests $\psi_i(C, C') = \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C')$.
- $|\psi_i| = 2\,|\psi_{i-1}|$, and so by induction, one can show that $|\psi_m| = 2^m$ which is exponential and is of no use.
- Now recall our earlier discussion on another representation of $\psi_i(C, C')$.

# The Proof continued: TQBF is PSPACE-hard

So, we have

$$\begin{aligned}\psi_i(C, C') &= \exists C'' \forall D_1 \forall D_2 ((\underbrace{D_1 = C} \wedge \underbrace{D_2 = C''}) \vee \\ &\quad (\underbrace{D_1 = C''} \wedge \underbrace{D_2 = C'})) \Rightarrow \psi_{i-1}(D_1, D_2)\end{aligned}$$

# The Proof continued: TQBF is PSPACE-hard

So, we have

$$\begin{aligned}\psi_i(C, C') &= \exists C''\forall D_1\forall D_2((\underbrace{D_1 = C}\wedge\underbrace{D_2 = C''})\vee \\ &\quad (\underbrace{D_1 = C''}\wedge\underbrace{D_2 = C'}))\Rightarrow \psi_{i-1}(D_1, D_2)\end{aligned}$$

- Here $|\psi_i| \leq |\psi_{i-1}| + O(m)$. . .

# The Proof continued: TQBF is PSPACE-hard

So, we have

$$\begin{aligned}\psi_i(C, C') &= \exists C''\forall D_1\forall D_2((\underbrace{D_1 = C} \wedge \underbrace{D_2 = C''}) \vee \\ &\quad (\underbrace{D_1 = C''} \wedge \underbrace{D_2 = C'})) \Rightarrow \psi_{i-1}(D_1, D_2)\end{aligned}$$

- Here $|\psi_i| \leq |\psi_{i-1}| + O(m)$. . .
- The above recursion solves to $|\psi_m| \leq O(m^2) = O(S(n)^2)$.

# The Proof continued: TQBF is PSPACE-hard

So, we have

$$\begin{aligned}\psi_i(C, C') &= \exists C''\forall D_1\forall D_2((\underbrace{D_1 = C} \wedge \underbrace{D_2 = C''}) \vee \\ &\quad (\underbrace{D_1 = C''} \wedge \underbrace{D_2 = C'})) \Rightarrow \psi_{i-1}(D_1, D_2)\end{aligned}$$

- Here $|\psi_i| \leq |\psi_{i-1}| + O(m)$. . .
- The above recursion solves to $|\psi_m| \leq O(m^2) = O(S(n)^2)$.
- For the sake of completeness, note that any QBF can be converted into its prenex normal form in polynomial time.

# The Proof continued: TQBF is PSPACE-hard

So, we have

$$\begin{aligned}\psi_i(C, C') &= \exists C''\forall D_1\forall D_2((\underbrace{D_1 = C}\wedge\underbrace{D_2 = C''})\vee \\ &\quad (\underbrace{D_1 = C''}\wedge\underbrace{D_2 = C'}))\Rightarrow \psi_{i-1}(D_1, D_2)\end{aligned}$$

- Here $|\psi_i| \leq |\psi_{i-1}| + O(m)$. . .
- The above recursion solves to $|\psi_m| \leq O(m^2) = O(S(n)^2)$.
- For the sake of completeness, note that any QBF can be converted into its prenex normal form in polynomial time.

**An interesting observation**

# The Proof continued: TQBF is PSPACE-hard

So, we have

$$\begin{aligned}\psi_i(C, C') &= \exists C''\forall D_1\forall D_2((\underbrace{D_1 = C} \wedge \underbrace{D_2 = C''}) \vee \\ &\quad (\underbrace{D_1 = C''} \wedge \underbrace{D_2 = C'})) \Rightarrow \psi_{i-1}(D_1, D_2)\end{aligned}$$

- Here $|\psi_i| \leq |\psi_{i-1}| + O(m)$. . .
- The above recursion solves to $|\psi_m| \leq O(m^2) = O(S(n)^2)$.
- For the sake of completeness, note that any QBF can be converted into its prenex normal form in polynomial time.

## An interesting observation

- Did the above proof anywhere assume that the outdegree of each vertex of $G_{M,x}$ is one?

# The Proof continued: TQBF is PSPACE-hard

So, we have

$$\psi_i(C, C') = \exists C'' \forall D_1 \forall D_2 ((\underbrace{D_1 = C} \wedge \underbrace{D_2 = C''}) \vee (\underbrace{D_1 = C''} \wedge \underbrace{D_2 = C'})) \Rightarrow \psi_{i-1}(D_1, D_2)$$

- Here $|\psi_i| \leq |\psi_{i-1}| + O(m)$. . .
- The above recursion solves to $|\psi_m| \leq O(m^2) = O(S(n)^2)$.
- For the sake of completeness, note that any QBF can be converted into its prenex normal form in polynomial time.

## An interesting observation

- Did the above proof anywhere assume that the outdegree of each vertex of $G_{M,x}$ is one?
- It did not, and hence, the proof holds for every $L \in$ NSPACE, i.e. $L \leq_P$ TQBF for every $L \in$ NSPACE. So, TQBF is also NSPACE-complete.

# Outline

# The Essence of PSPACE

- 'YES' answers of NP-complete problems have short (polynomial) certificates.

# The Essence of PSPACE

- 'YES' answers of NP-complete problems have short (polynomial) certificates.
- The analogy for PSPACE-complete problems is a winning strategy for a two-player game with perfect information, as in chess, where players make alternate moves.

# The Essence of PSPACE

- 'YES' answers of NP-complete problems have short (polynomial) certificates.
- The analogy for PSPACE-complete problems is a winning strategy for a two-player game with perfect information, as in chess, where players make alternate moves.
- What is a winning strategy for the first player?

# The Essence of PSPACE

- 'YES' answers of NP-complete problems have short (polynomial) certificates.
- The analogy for PSPACE-complete problems is a winning strategy for a two-player game with perfect information, as in chess, where players make alternate moves.
- What is a winning strategy for the first player?
- The first player has a winning strategy iff $\exists$ a 1st move for Player 1, s.t. $\forall$ possible 1st move of Player 2, $\exists$ a 2nd move for Player 1 s.t. ... Player 1 wins at the end.

# The Essence of PSPACE

- 'YES' answers of NP-complete problems have short (polynomial) certificates.
- The analogy for PSPACE-complete problems is a winning strategy for a two-player game with perfect information, as in chess, where players make alternate moves.
- What is a winning strategy for the first player?
- The first player has a winning strategy iff $\exists$ a 1st move for Player 1, s.t. $\forall$ possible 1st move of Player 2, $\exists$ a 2nd move for Player 1 s.t. ... Player 1 wins at the end.
- The problem of deciding whether the 1st Player has a winning strategy seems to require searching the tree of all possible moves. This was the case also for problems in NP.

# The Essence of PSPACE

- 'YES' answers of NP-complete problems have short (polynomial) certificates.
- The analogy for PSPACE-complete problems is a winning strategy for a two-player game with perfect information, as in chess, where players make alternate moves.
- What is a winning strategy for the first player?
- The first player has a winning strategy iff $\exists$ a 1st move for Player 1, s.t. $\forall$ possible 1st move of Player 2, $\exists$ a 2nd move for Player 1 s.t. ... Player 1 wins at the end.
- The problem of deciding whether the 1st Player has a winning strategy seems to require searching the tree of all possible moves. This was the case also for problems in NP.
- But, here, unlike NP, we have no short certificates.

# The QBF Game

- The 'board' for the QBF game is a boolean formula $\varphi$ whose free variables are $x_1, x_2, \ldots, x_{2n}$. The 1st player will pick values for $x_1, x_3, \ldots$ and the 2nd player will pick values for $x_2, x_4, \ldots$.

# The QBF Game

- The 'board' for the QBF game is a boolean formula $\varphi$ whose free variables are $x_1, x_2, \ldots, x_{2n}$. The 1st player will pick values for $x_1, x_3, \ldots$ and the 2nd player will pick values for $x_2, x_4, \ldots$.
- Player 1 wins iff at the end $\varphi(x_1, x_2, \ldots, x_{2n})$ is TRUE.

# The QBF Game

- The 'board' for the QBF game is a boolean formula $\varphi$ whose free variables are $x_1, x_2, \ldots, x_{2n}$. The 1st player will pick values for $x_1, x_3, \ldots$ and the 2nd player will pick values for $x_2, x_4, \ldots$.
- Player 1 wins iff at the end $\varphi(x_1, x_2, \ldots, x_{2n})$ is TRUE.
- Player 1 has a winning strategy if Player 1 has a way to win $\forall$ possible sequences of moves by Player 2, i.e.

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \ldots \exists x_{2n-1} \forall x_{2n} \varphi(x_1, x_2, \ldots, x_{2n})$$

# The QBF Game

- The 'board' for the QBF game is a boolean formula $\varphi$ whose free variables are $x_1, x_2, \ldots, x_{2n}$. The 1st player will pick values for $x_1, x_3, \ldots$ and the 2nd player will pick values for $x_2, x_4, \ldots$.
- Player 1 wins iff at the end $\varphi(x_1, x_2, \ldots, x_{2n})$ is TRUE.
- Player 1 has a winning strategy if Player 1 has a way to win $\forall$ possible sequences of moves by Player 2, i.e.

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \ldots \exists x_{2n-1} \forall x_{2n} \varphi(x_1, x_2, \ldots, x_{2n})$$

- Player 1's winning strategy is nothing but the above QBF being TRUE.

# The QBF Game

- The 'board' for the QBF game is a boolean formula $\varphi$ whose free variables are $x_1, x_2, \ldots, x_{2n}$. The 1st player will pick values for $x_1, x_3, \ldots$ and the 2nd player will pick values for $x_2, x_4, \ldots$.
- Player 1 wins iff at the end $\varphi(x_1, x_2, \ldots, x_{2n})$ is TRUE.
- Player 1 has a winning strategy if Player 1 has a way to win $\forall$ possible sequences of moves by Player 2, i.e.

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \ldots \exists x_{2n-1} \forall x_{2n} \varphi(x_1, x_2, \ldots, x_{2n})$$

- Player 1's winning strategy is nothing but the above QBF being TRUE.
- So, it turns out that deciding whether Player 1 has a winning strategy for a given board in the QBF game is PSPACE-complete.