

Definition 8.1

- Let M be a deterministic Turing machine, DTM, that halts on all inputs. The space complexity of M is the function $f : \mathcal{N} \rightarrow \mathcal{N}$, 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 .

22c:131 Limits of Computation

Hantao Zhang

<http://www.cs.uiowa.edu/~hzhang/c131>

The University of Iowa
Department of Computer Science

Ch.8 Space Complexity

Example 1 (8.3)

SAT can be solved with the linear space algorithm M_1 :

$M_1 =$ "On input $\langle \Phi \rangle$, where Φ is a Boolean formula:

1. For each truth assignment to the variables x_1, \dots, x_n of Φ :
 - (a) Evaluate Φ on that truth assignment;
 - (b) If Φ evaluates to 1, *accept*.
2. If Φ never evaluates to 1, *reject*".

Estimation of space complexity

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

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

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* ^{q} .

Example 2 (8.4)

Testing whether an NFA accepts all strings,

$$ALL_{NFA} = \{\langle A \rangle \mid A \text{ is an NFA} \wedge L(A) = \Sigma^*\}$$

$$\overline{ALL_{NFA}} = \{\langle A \rangle \mid A \text{ is an NFA} \wedge L(A) \neq \Sigma^*\}$$

We show that $\overline{ALL_{NFA}} \in NSPACE$.

Proof idea: construct N , a nondeterministic linear space algorithm that decide the complement $\overline{ALL_{NFA}}$ 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 $NTIME(f(n)) \subseteq TIME(2^{O(f(n))})$.

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

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 $CANYIELD(c_1, c_2, t)$, 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.

SPACE vs NSPACE

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

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

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.

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 $2^{df(n)}$ configurations.

Then the DTM M is:

$M =$ "On input w

1. Let c_s be q_0w , where q_0 is the start state of N .
2. Output the result of $CANYIELD(c_s, c_a, 2^{df(n)})$.

$CANYIELD(c_1, c_2, t)$

$CANYIELD(c_1, c_2, t)$ 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.

$CANYIELD =$ "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* if both fail.
2. If $t > 1$ then for each configuration of N on w using space $f(n)$:
 - (a) Run $CANYIELD(c_1, c_m, t/2)$.
 - (b) Run $CANYIELD(c_m, c_2, t/2)$.
 - (c) If both 2(a) and 2(b) accept then *accept*.
3. If haven't yet accepted at 2(c), *reject*".

The class $PSPACE$

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

$$PSPACE = \cup_k SPACE(n^k)$$

$$NPSPACE = \cup_k NSPACE(n^k)$$

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

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

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 $O(f^2(n))$

PSPACE-Completeness

A language B is *PSPACE*-complete if it satisfies two conditions:

1. $B \in PSPACE$
2. Every $A \in PSPACE$ is polynomially time reducible to B .

If B satisfies only 2, B is said to be *PSPACE*-hard.

The class PSPACE

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

$$PSPACE = \cup_k SPACE(n^k)$$

$$NPSPACE = \cup_k NSPACE(n^k)$$

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

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

SUMMARY $P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$

TQBF is PSPACE-complete

The following polynomial space algorithm decides *TQBF*:

$T =$ "On input $\langle \Phi \rangle$, a fully quantified Boolean formula:

1. If Φ contains no quantifiers, then it is an expression with only constants. So, evaluate Φ and *accept* if it is true; otherwise *reject*.
2. If $\Phi = \exists x \Psi$, recursively call T on Ψ , first with 0 substituted for x and then with 1 substituted for x . If either result is *accept*, *accept*; otherwise *reject*.
3. If $\Phi = \forall x \Psi$, recursively call T on Ψ , first with 0 substituted for x and then with 1 substituted for x . If both results are *accept*, *accept*; otherwise *reject*".

A PSPACE-complete language

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

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

TQBF is *PSPACE*-complete.

Proof Review of any $A \leq_P \text{SAT}$

- On input $\langle A, w \rangle$, f produces Φ_w
- Variables of Φ_w :** Let $N = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, and $C = Q \cup \Gamma \cup \{\#\}$. For each $1 \leq i, j \leq n^k \wedge s \in C$ we have a variable $x_{i,j,s}$ in Φ_w .
- Cells:** 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 Φ_w is $\Phi_w = \Phi_{\text{cell}} \wedge \Phi_{\text{start}} \wedge \Phi_{\text{move}} \wedge \Phi_{\text{accept}}$.

Proof

Use T and the construction of CANYIELD to prove that any PSPCE problem reduces to TQBF in polynomial time.

An accepting tableau

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

- Φ_{start} ensures that the first row of the tableau is the starting configuration of N on w by the equality:

$$\Phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,w_1} \wedge \dots \wedge x_{1,n+3,w_n} \wedge \dots \wedge x_{1,n^k-1,\square} \wedge x_{1,n^k,\#}$$
- Φ_{accept} guarantees that an accepting configuration occurs in the tableau by placing q_a in one of the cells by: $\Phi_{\text{accept}} = \bigvee_{1 \leq i, j \leq n^k} x_{i,j,q_a}$
- Φ_{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:

- at least one variable that is associated with a cell is on, by: $\bigvee_{s \in C} x_{i,j,s}$
- no more than one variable is on for each each cell, by:

$$\bigwedge_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}})$$

Thus, any satisfying assignment specifies one symbol in each cell by:

$$\Phi_{\text{cell}} = \bigwedge_{1 \leq i, j \leq n^k} [(\bigvee_{s \in C} x_{i,j,s}) \wedge (\bigwedge_{s \neq t \in C} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}))]$$

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 Φ 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 $\Phi_{c_{start}, c_{accept}, h}$, where $h = 2^{df(n)}$ for a constant d chosen so that M has no more than $2^{df(n)}$ configurations. Assume that t is a power of 2.

Legal window

A 2×3 window of cells is legal if that window does not violate the actions specified by N 's transition function. To explain, consider the transitions: $\delta(q_1, a) = \{q_1, b, R\}$, $\delta(q_1, b) = \{q_2, c, L\}$, (q_2, a, R) . Examples of legal windows for this machine are in Figure 1:

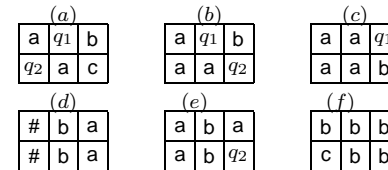


Figure 1: Examples of legal windows

A Formula Game

- A formula game is a pair (X, Ψ) , where X is a list of boolean variables, $X = [x_1, x_2, \dots, x_m]$, and Ψ 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 Ψ 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 $\mathcal{O}(n^k)$ variables.
- For $t = 1$ formula 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:
 $\Phi_{c_1, c_2, t} = \exists m_1 [\Phi_{c_1, m_1, t/2} \wedge \Phi_{m_1, c_2, t/2}]$ where m_1 is a configuration of M . $\exists m_1$ is shorthand for $\exists x_1, x_2, \dots, x_l, l = \mathcal{O}(n^k)$ and x_1, \dots, x_l 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 $\forall x \in \{y, z, \dots\}$ denotes $\forall x [x = y \vee x = z \vee \dots]$.

TQBF vs Formula Game

- **Theorem:** Given a game (X, Ψ) , suppose E moves first and m is even. Let $\Phi = \exists x_1 \forall x_2 \exists x_3 \dots \forall x_m [\Psi]$. Then Φ is true iff E has a winning strategy for the game (X, Ψ) .
- **Lemma:** For any fully quantified boolean formula Φ , there exists an equivalent fully quantified boolean formula Φ' in prenex normal form whose quantifiers alternate between \exists and \forall .
- **Theorem:** For any fully quantified boolean formula Φ , there exists a formula game (X, Ψ) such that player E has a winning strategy iff Φ is true.

Example of Formula Game

Suppose E moves first.

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

$FORMULA-GAME = \{ \langle X, \Phi \rangle \mid \text{Player } E \text{ has a winning strategy in the formula game } (X, \Phi) \}$ is *PSPACE*-complete.

GG is in PSPACE

The following algorithm M decides whether player I has a winning strategy for GG game:

$M =$ "On input $\langle G, b \rangle$ where G is a directed graph and b is a node of G :

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

The GG problem

$GG = \{ \langle G, b \rangle \mid \text{Player I has a winning strategy for the generalized geography game played on graph } G \text{ starting at node } b \}$.

Theorem 8.14 GG is $PSPACE$ -complete

GG is PSPACE-hard

Theorem: Formula-Game $\leq_P GG$.

- Let (X, Φ) be a formula game, where Φ 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, Φ) for formula game iff player I has a winning strategy in (G, b) in GG .
- $V = V_1 \cup V_2$, where
 - $V_1 = \{b_i, x_i, \bar{x}_i, e_i \mid 1 \leq i \leq k, k = |X|\}$ (assume $|X|$ is odd), and
 - $V_2 = \{c_j, c_{j,i} \mid 1 \leq j \leq m, 1 \leq i \leq k, c_j = (c_{j,1} \vee \dots \vee c_{j,k})\}$.
- $E = E_1 \cup E_2 \cup E_3$, where
 - E_1 constructs a chain of "dimonds" among V_1 ,
 - E_2 constructs a tree among V_2 ,
 - E_3 connects V_1 and V_2 .
- $b = b_1$