CS5371
Theory of Computation

Lecture 24: Complexity IX
(PSPACE-complete, L, NL,
NL-complete)

Objectives

- PSPACE-complete languages + examples
- The classes L and NL
- NL-complete languages + examples
- Proof of Savitch's Theorem

PSPACE-complete

Definition: A language B is PSPACE-complete if it satisfies the two conditions below:

1. B is in PSPACE
2. Every other language in PSPACE can be polynomial time reducible to B

If B is just satisfies Condition 2, we say B is PSPACE-hard

Question: Why don't we use polynomial space reducible?

Quantified Boolean Formula

- Mathematical statements usually involve quantifiers: $\forall$ (for all) and $\exists$ (there exists)
  - E.g., $\forall x\ F(x)$ means for every value of x, the statement F(x) is TRUE
  - E.g., $\exists x\ F(x)$ means there exists some value of x such that F(x) is TRUE
- Boolean formulas with quantifiers are called quantified Boolean formulas
  - E.g., $\exists y\ (\ y = x+1)$ and $\forall x\ (\exists y\ (\ y > x))$ are quantified Boolean formulas

Quantified Boolean Formula (2)

- The scope of a quantifier is the fragment of statement that appears within the matched parentheses following the quantified variable
  - E.g., the scope of $\exists y$ in $\exists y(y = x+1)$ is $(y = x+1)$
- If each variable in a formula appears within the scope of some quantifier, the formula is said to be fully quantified
  - A fully quantified Boolean formula is always either TRUE or FALSE

TQBF is PSPACE-complete

Let TQBF be the language

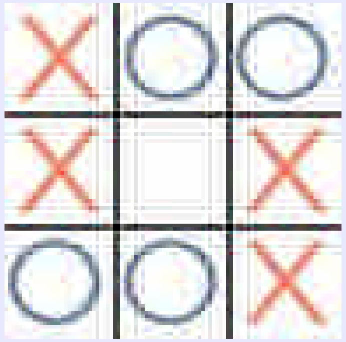
$\{\langle F \rangle \mid F$ is a true fully quantified Boolean formula $\}$

Theorem: TQBF is PSPACE-complete.

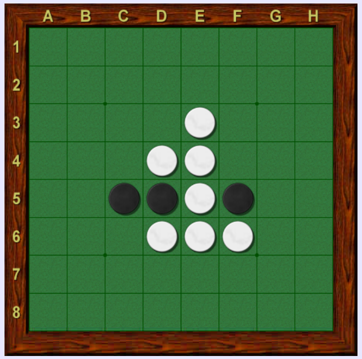
PSPACE-complete [more examples]

The generalized version of some common games we play are PSPACE-complete:

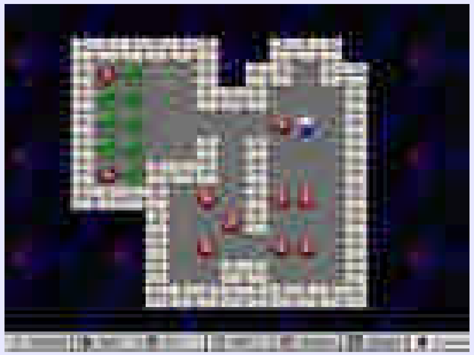
Tic-Tac-Toe



Reversi



Sokoban



Two-tape TM

- We now introduce a TM with two tapes:
  1. A read-only input tape
  2. A read/write working tape
- For this TM to operate, the input tape head always remain on the portion of the tape containing the input
- The space complexity of an algorithm is now the number of working tape cells used
  - This is the same as before if space complexity is at least linear

The Classes L and NL

Definition:

1. L is the class of languages that are decidable in logarithmic space on a two-tape DTM. In other words,

$$L = SPACE(\log n)$$

2. NL is the class of languages that are decidable in logarithmic space on a two-tape NTM. In other words,

$$NL = NSPACE(\log n)$$

Example Language in L

Let A be the language

$$\{0^k 1^k \mid k > 0\}$$

Theorem: A is in L.

Example Language in NL

Let PATH be the language

$\{ \langle G,s,t \rangle \mid G$ is a directed graph that has a directed path from s to $t \}$

Theorem: PATH is in NL.

Log space Reducible

- A log space transducer is a DTM with a read-only input tape, a write-only output tape, and an $O(\log n)$-cell read/write work tape.
- A log space transducer M computes a function $f: \Sigma^* \rightarrow \Sigma^*$ where $f(w)$ is the string remaining on output tape after M halts when it is started with w on its input tape
- We call f a log space computable function

Log space Reducible (2)

Definition: A language A is log space reducible to a language B, written as $A \leq_L B$, if some log space computable function f exists such that

$$w \text{ in } A \Leftrightarrow f(w) \text{ in } B$$

Properties of log space reducible

Theorem: If $A \leq_L B$ and $B \in L$, then $A \in L$.

Proof Idea: To show that "deciding whether an input w is in A or not" can be done in O(log n) space, a tempting approach is to perform log space reduction from A to B, obtaining f(w) and then decide if f(w) is in B or not…

Properties of log space reducible (2)

… Unfortunately, $f(w)$ may be very long, so that the overall space usage is not in $O(\log n)$.

However, observe that the decider for B, say M_B, does not need to have all $f(w)$ stored in the input tape, as long as when M_B needs to read a particular character, the character is ready for it to read → This can be done by a TM M_A that uses $O(\log n)$ space only

Properties of log space reducible (3)

... Now, to decide if w is in A, we make use of TM M_A and M_B. The total space required is $O(\log n)$ for M_A and $1+ \log |f(w)|$ for M_B.

It remains to bound the value of $|f(w)|$.

Since $f(w)$ is generated from a log space transducer which halts from all inputs, $|f(w)|$ is at most the maximum number of configurations this transducer can use, which is $|w|2^{O(\log |w|)}$ $\rightarrow$ space required for $M_B = 1+ \log |f(w)| = O(\log |w|) = O(\log n)$

NL-complete

Definition: A language B is NL-complete if it satisfies the two conditions below:

1. B is in NL
2. Every other language in NL can be log space reducible to B

Corollary: If any NL-complete language is in L, then L = NL.

Question: Why don't we use polynomial time reducible?

PATH is NL-complete

Theorem: PATH is NL-complete.

Proof: See Chapter 8.5 (page 325)

Corollary: NL $\subseteq$ P

Proof: Any language is log space reducible to PATH. Thus, the reducer uses O(log n) space, so it runs in polynomial time. Also, PATH is in P. This completes the proof.

PATH is coNL

Theorem: PATH is coNL.

Proof: See Chapter 8.6 (page 327)

Corollary: NL = coNL

Proof: We show that (1) NL $\subseteq$ coNL and (2) coNL $\subseteq$ NL. (see next slide)

NL = coNL

(1) For any $x \in NL$, we have $x \leq_L$ PATH since PATH is NL-complete. Then by the same reduction function, we have $x' \leq_L$ PATH'. Thus, $x' \in NL$ since PATH' is in NL, so $x \in coNL$. Thus, $NL \subseteq coNL$

(2) For any $x \in coNL$, $x' \in NL$. Similarly, we have $x' \leq_L$ PATH and $x \leq_L$ PATH'. Again, since PATH' is in NL, so that $x \in NL$. Thus, $coNL \subseteq NL$.

Summary

$$L \subseteq NL = coNL \subseteq P \subseteq PSPACE$$