Chapter 8, Part 1

PSPACE

PSPACE

PSPACE is the class of all languages decided by polynomial space Turing machines.

NP is included in PSPACE.

Savitch's Theorem

Let $p(n) = n^k + k$ be a polynomial. Let A be decided by a $p(n)$ space bounded nondeterministic Turing machine M. We can assume that on input x (1) M first marks its first $p(|x|)$ tape cells (thus M uses exactly $p(|x|)$ cells) and (2) before accepting x M replaces all the marked cells with blanks and moves its head to the leftmost cell (thus M has a unique accepting configuration).

Reachability

There is an integer d such that for all n the number of possible configurations of M on input having length n is at most $2^{dp(n)}$.

For two configurations C and C' of M and an integer i, write $R(C, C', i)$ if C' can be reached from C within 2^i steps.

Let x be an input having length n. Let C_{ini} be the initial configuration of M on input x and let C_{acc} be the unique accepting configuration of M on inputs of length n.

Then M accepts x if and only if $R(C_{ini}, C_{acc}, dp(n))$ is true.

Recursive Test for R

Computing the value of $R(C, D, i)$.

Step 1 If $i = 0$, return true if either $C = D$ or (D is one of the next possible configurations of C); otherwise, return false.

Step 2 If $i > 0$, do the following for each possible configuration E of M that uses $p(n)$ cells:

Step 2a Compute $f = R(C, E, i - 1)$.

Step 2b If f is true, compute $g = R(E, D, i - 1)$ and if g is true, return true.

Step 3 Return false.

Analysis

The algorithm is deterministic.

To test whether $x \in L(M)$, the value i starts from $dp(n)$, so the recursion depth is $dp(n)$. Thus, the space requirement is $O(p(n)^2)$.

This implies: **Savitch's Theorem.** **NPSPACE = PSPACE.**

Formula Representation

For $i > 0$, $R(C, D, i) =$ true if and only if there exists a configuration E such that $R(C, E, i - 1) = R(E, D, i - 1) =$ true. This can be rewritten as:

$R(C, D, i) =$ true if and only if there exists a configuration E such that for all configurations F and G, if $(F = C \wedge G = E) \vee (F = E \wedge G = D)$ then $R(F, G, i - 1) =$ true.

This is equivalent to: $R(C, D, i) =$ true if and only if

- there exists a configuration E such that
- for all configurations F and G,
- $((F \neq C \vee G \neq E) \wedge (F \neq E \vee G \neq D)) \vee$ $R(F, G, i - 1) =$ true.

Formula Unfolding

If $i \geq 2$, $R(C, D, i) =$ true if and only if

- there exists a configuration E such that
- for all configurations F and G,
- there exists a configuration E' such that
- for all configurations F' and G',
- $((F \neq C \vee G \neq E) \wedge (F \neq E \vee G \neq D)) \vee$ $((F' \neq F \vee G' \neq E') \wedge (F' \neq E' \vee G' \neq G)) \vee$ $R(F', G', i-2) =$ true.

Formula Unfolding

If $i \geq 3$, $R(C, D, i) =$ true if and only if

- there exists a configuration E such that
- for all configurations F and G,
- there exists a configuration E' such that
- for all configurations F' and G',
- there exists a configuration E'' such that
- for all configurations F'' and G'',
- $((F \neq C \vee G \neq E) \wedge (F \neq E \vee G \neq D)) \vee$
  $((F' \neq F \vee G' \neq E') \wedge (F' \neq E' \vee G' \neq G)) \vee$
  $((F'' \neq F' \vee G'' \neq E'') \wedge (F'' \neq E'' \vee G'' \neq G')) \vee$
  $R(F'', G'', i-3) =$ true.

Repetition

Using the boolean encoding of configurations as we used to prove NP-completeness of SAT, R can be expressed as a boolean formula. So, $x \in L(M)$ can be expressed as:

- there exists an assignment to a set of variables S_1 such that,
- for all assignments to a set of variables T_1,
- there exists an assignment to a set of variables S_2 such that,
- for all assignments to a set of variables T_2,
- $\cdots$
- there exists an assignment to a set of variables $S_{dp(n)}$ such that,
- for all assignments to a set of variables $T_{dp(n)}$,
- formula ϕ is true,

where the variables of ϕ are $S_1 \cup T_1 \cup \cdots \cup S_{dp(n)} \cup T_{dp(n)}$.

TQBF

A **quantified formula** is one in which **quantifiers** $\exists$ and $\forall$ may appear. Each quantifier should be followed immediately by a variable, which is **bound** to the quantifier. $\exists x$ means "for some value of x" and $\forall x$ means "for every value of x."

The quantification applies to every occurrence of the variable to the right of the point of quantification within the innermost pair of parentheses that contains the quantifier. This is called the **scope** of the quantifier.

A formula is **fully quantified** if all the variables are bound.

A formula is in **prenex normal form** if all of its quantifiers appear at the beginning.

$TQBF = \{\phi \mid \phi$ is a true fully quantified Boolean formula $\}$.

Recap

From the previous discussion, we have that every language in PSPACE is reducible to $TQBF$.

Recap

From the previous discussion, we have that every language in PSPACE is reducible to $TQBF$.

It is easy to show that $TQBF$ is in PSPACE. Thus, we have

Theorem. **$TQBF$ is PSPACE-complete.**

Recap

From the previous discussion, we have that every language in PSPACE is reducible to $TQBF$.

It is easy to show that $TQBF$ is in PSPACE. Thus, we have
Theorem. **$TQBF$ is PSPACE-complete.**

The formula ϕ that appears in the mapping reduction can be easily converted to a 3CNF formula, so:

Theorem. **A version of $TQBF$ in which the base formulas must be 3CNF is PSPACE-complete.**

Formula Game

Suppose ϕ is a Boolean formula over $x_1, \ldots, x_k$ without quantifiers. Consider the game played by two players in which they take turns, starting from Player 1, in assigning values to the variables $x_1, x_2, \cdots, x_k$ in this order. Player 1 wins if the formula evaluates to TRUE for the assignment and Player 2 wins otherwise.

Define $\mathit{FORMULA\text{-}GAME} = \{\phi \mid \text{Player 1 can always win for } \phi\}$.

Formula Game is PSPACE-complete

The condition $\phi \in FORMULA\text{-}GAME$ is rewritten as:

$$(\exists x_1)(\forall x_2) \cdots (Q_k x_k)\phi,$$

where the quantifiers alternate. This is essentially a variation of $TQBF$.

On the other hand, in the reduction from the arbitrary PSPACE-language to $TQBF$, the formula can be modified so that the quantifiers alternate by inserting dummy variables with appropriate quantifiers.

General Geography

Given a directed graph $G = (V, E)$ with a specified node s, consider the following game played by two players:

- Initially, $x = s$ and $W = \{s\}$.
- Then the players take turns, beginning with the move by Player 1, in selecting a node $u \notin W$ such that $(x, u) \in E$, adding u to W, and setting x to u.
- A player wins if the other cannot make a move.

This is called **General Geography**.

Define $GG = \{(G, s) \mid (G, s)$ is an instance of General Geography such that Player 1 has a winning strategy$\}$.

General Geography is PSPACE-complete

$GG \in \mathbf{PSPACE}$. Let (G, s) be an instance of GG of some n nodes. Consider the following procedure **Search**:

- Input (G, x, W, b), $x \in V$, $W \subseteq V$, $b \in \{1, 2\}$.
- Let $u_1, \ldots, u_k$ be the nodes not in W to which there is an arc from x.
- Output "no" if there is no such node.
- For each i, $1 \leq i \leq k$, compute the output of Search on input $(G, u_i, W \cup x$,3-b$)$.
- If the output is "no" for some i return "yes"
- Otherwise, return "no".

We have only to call Search on $(G, s, \emptyset, 1)$.

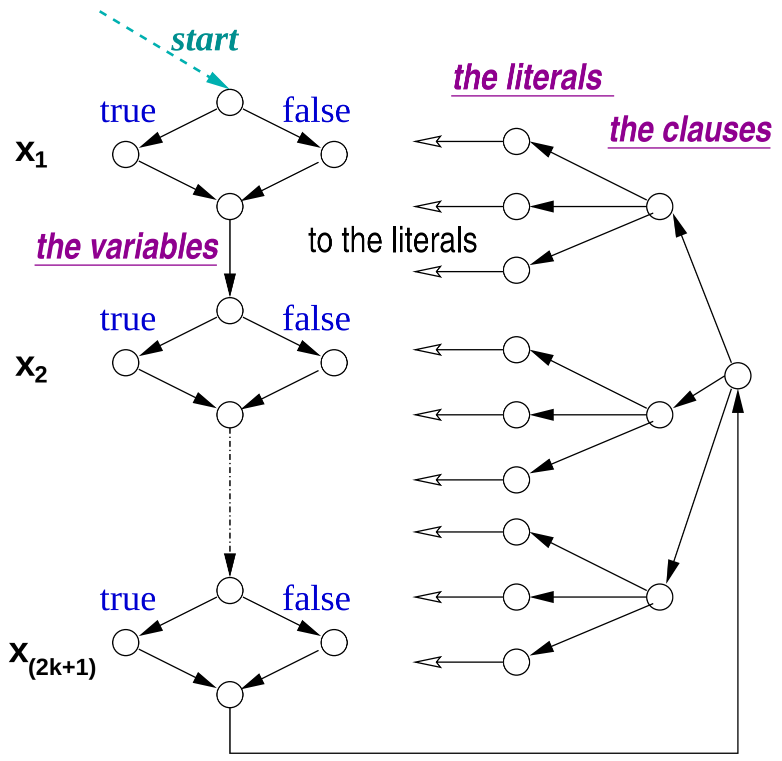
GG is PSPACE-complete

The recursion depth is the number of nodes of the graph and the number of branches is bounded by the number of nodes.

Thus, a polynomial space machine can execute it.

General Geography is PSPACE-Complete

Consider the version of *FORMULA-GAME* in which the formula is restricted to a 3CNF with an odd number of variables. Reduce the formula to an instance of *GG* as follows:


start
true
false
x_1
the variables
true
false
x_2
true
false
$x_{(2k+1)}$
to the literals
the literals
the clauses