Tutorial 11

Gidon Rosalki

2025-06-11

1 Space complexity

Definition 1.1 (Space complexity for a TM run)**.** *Let there be a TM M, and $w \in \Sigma^*$. We denote r and l to be the rightmost and leftmost cells that get used in the run of M on w, respectively. We will say that M runs on w in space*

$$r - l + 1$$

Definition 1.2 (Space complexity for an NTM run)**.** *Let there be an NTM N, and $w \in \Sigma^*$, such that all branches of N on w halt. We denote r_a and l_a to be the rightmost and leftmost cells that get used in the run of N on w, respectively. We will say that N runs on w in space*

$$\max_{a \text{ is a run of } N \text{ on } w} \{r_a - l_a + 1\}$$

Definition 1.3 (Space complexity)**.** *Let T be a TM or NTM wherein all branches halt on all inputs, the space complexity of T is the function*

$$g : \mathbb{N} \to \mathbb{N}$$

where for every $n \in \mathbb{N}$, $g(n)$ is the maximum space that T uses on any word of length n. For a function $f : \mathbb{N} \to \mathbb{N}$, we say that T runs in space

$$O(f(n)) \text{ if } g(n) = O(f(n))$$

Definition 1.4 (Space classes)**.** *For every $f : \mathbb{N} \to \mathbb{N}$*

$$SPACE(f(n)) = \{L \subseteq \Sigma^* : L \text{ can be decided by a TM that runs in } O(f(n)) \text{ space}\}$$

and

$$NSPACE(f(n)) = \{L \subseteq \Sigma^* : L \text{ can be recognised by an NTM that runs in } O(f(n)) \text{ space}\}$$

Definition 1.5 (PSPACE and NPSPACE)**.** *Similarly to P and NP, we define*

$$PSPACE = \bigcup_{k=1}^{\infty} SPACE\left(n^k\right)$$

$$NPSPACE = \bigcup_{k=1}^{\infty} NSPACE\left(n^k\right)$$

2 TQBF

A **quantified** boolean formula is a boolean formula that is preceded by $\exists$ and $\forall$ quantifiers on the variables. A **fully** quantified formula is one such that every variable is under the scope of a quantifier. A fully quantified formula is always either true or false.

Example 1.

$$\forall x\ \exists y\ \ ((x \vee y) \wedge (\overline{x} \vee \overline{y}))$$

Solution. This formula is true, since for $x = F$ we can take $y = T$, and for $x = T$ we can take $y = F$. □

Example 2.

$$\exists y\ \forall x\ \ ((x \vee y) \wedge (\overline{x} \vee \overline{y}))$$

Solution. This is false, since for both $y = F$ and for $y = T$, we can find a value for x, (namely $x = y$), such that one of the conjuncts does not hold. □

Definition 2.1 (TQBF)**.**

$$TQBF = \{\langle\varphi\rangle : \varphi \text{ is a true fully quantified boolean formula}\}$$

If such a formula only contains $\exists$ quantifiers for all the variables, then this is a satisfiability problem, so this language is sometimes called QSAT (quantified SAT)

Theorem 1 (TQBF is in PSPACE)**.**

$$TQBF \in PSPACE$$

Proof . We will show this by constructing a TM that runs a recursive algorithm.
Construction: On input $\langle\varphi\rangle$, a fully quantified boolean formula:

1. If φ contains no quantifiers, meaning it is only made of boolean constants, then we will evaluate it, and accept if it evaluated to true, and reject if it evaluated to false.
2. If $\varphi = \exists x\varphi'$, then evaluate φ', first with $x = F$, and then with $x = T$. If **either** returns true, then accept, and if both are false, then reject.
3. If $\varphi = \forall x\varphi'$, then evaluate φ', first with $x = F$, and then with $x = T$. If **both** return true, then accept, and if either are false, then reject.

Correctness: It can be shown by induction that M decides TQBF.
Space complexity: Observe, the depth of recursion is at most the number of variables m. We make at most 2 recursive calls per quantifier, and at each level of recursion, we need to store the formula with the replaced variables, so the total space used is $O(n)$. At the bottom of the recursion, we need to evaluate the formula, which takes another $O(n)$ space. Therefore, T runs in $O(m \cdot n)$, and since $m = O(n)$, we get a total space $O\left(n^2\right)$. □

3 Encoding configurations using boolean formulas

To prove that TQBF is PSPACE-complete, we need to give some background on how to encode configurations of a Turing-machine using Boolean formulas. This section might be a bit cumbersome, but is wonderfully useful.
Let M be a TM, and let s be a tape size. We encode a configuration that uses at most s tape cells using the following assignment of boolean variables:

- **Machine state**: For each $q \in Q$, we will define z_q such that $z_q = T$ **if and only if** the machine is in state q
- **Head position**: For each $i \in [s]$, we will define y_i such that $y_i = T$ **if and only if** the head is over the ith cell
- **Tape content**: For each $i \in [s]$, and $a \in \Gamma$, we have a variable $x_{i,a}$, such that $x_{i,a} = T$ **if and only if** a is written in the ith cell

We will denote a tuple of all these variables c, and claim the following theorem:

Theorem 2. *1. There exists a boolean formula $\varphi_{valid}(c)$, that evaluates to True **if and only if** c is a valid encoding of a configuration*

*2. There exists a boolean formula $\varphi(c_1, c_2)$ that evaluates to True **if and only if** c_2 is a consecutive configuration to c_1.*

We can construct both formulas in polynomial time with respect to s

Proof . 1. Let us define

$$\varphi_{\text{valid}}(c) = \bigwedge_{i\in[s]} \bigvee_{a\in\Gamma} \left(x_{i,a} \wedge \bigwedge_{b\in\Gamma\setminus a} \overline{x_{i,b}} \right) \wedge \bigvee_{i\in[s]} \left(y_i \wedge \bigwedge_{j\in[s]\setminus\{i\}} \overline{y_j} \right) \wedge \bigvee_{q\in Q} \left(z_q \wedge \bigwedge_{r\in Q\setminus\{w\}} \overline{z_r} \right)$$

Where the first brackets indicate that each tape cell holds exactly one letter, the second that the head is at exactly one tape position, and the third that the machine is in exactly one state. Note that the formula is of length $O\left(s^2\right)$, (Q and Γ are constants).

2. For every $q \in Q$, and $a \in \Gamma$, if $\delta(q, a) = (r, b, R)$, we define for every $i \in [s]$

$$\varphi_{i,a,q}(c_1, c_2) = \left(x^1_{i,a} \wedge y^1_i \wedge z^1_q\right) \implies \left(x^2_{i,b} \wedge y^2_{i+1} \wedge z^2_r \wedge \bigvee_{j\in[s]\setminus\{i\}} \bigvee_{d\in\Gamma} \left(x^1_{j,d} \Leftrightarrow x^2_{j,d}\right) \right)$$

and if $\delta(q, a) = (r, b, L)$, then we do the exact same thing, but with $i - 1$. The formula (for the R case) guarantees that if in c_1 the head is in position i over the letter a, and the machine is in state q, then in c_2 the head is in position $i + 1$ over the letter b and the machine is in state r. The first part checks the previous configuration and the second part verifies that what should be changed is changed, and what should not be changed remains as is.
There are several exceptions in the definition of $\varphi_{i,a,q}$:

(a) If the head is in the leftmost cell, and the move is L, then the head does not move

(b) If the head is the rightmost cell, and the move is R, then there is no valid consecutive configuration

(c) If the configuration is accepting, then we require that the configuration stays the same.

Finally, let us define:

$$\varphi(c_1, c_2) = \varphi_{\text{valid}}(c_1) \wedge \varphi_{\text{valid}}(c_2) \wedge \bigvee_{i \in [s]} \bigvee_{a \in \Gamma} \bigvee_{q \in Q} \varphi_{i,a,q}(c_1, c_2)$$

Note, the length of the formula is $O\left(s^2\right)$. A similar construction also works for NTMs.

□

Definition 3.1 (PSPACE-hard)**.** *A language L is PSPACE-hard if for every $K \in PSPACE$ it holds that $K \leq_p L$*

Theorem 3 (TQBF is PSPACE-hard)**.** *Proof .* We show that $\forall L \in PSPACE\ L \leq_p TQBF$. Let $L \in PSPACE$, and let M be a machine that decides L, suing at most $s(n)$, where s is some polynomial. Assume w.l.o.g. that M has a single accepting configuration on each word w (We do not lose generality since every machine in PSPACE has such an equivalent machine also in PSPACE, that is obtained by changing every accepting state to a state that deletes the contents of the tape before accepting).
Given a word w, our reduction outputs a formula η, such that η has the value true **if and only if** M accepts w. We can do this by using the tools developed earlier. Let $n = |w|$, $s = s(n)$, c_0 be the start configuration, c_{acc} be the accepting configuration, and t be the maximum length of a run of a TM in space s.

Naïve attempt: Let us define

$$\exists c_1, \ldots, c_t \left(c_1 = c_0 \wedge (c_t = c_{\text{acc}}) \wedge \bigwedge_{i=1}^{t-1} \varphi(c_i, c_{i+1}) \right)$$

The problem here is that the maximum run time t is exponential in s, and thus in n, and thus this would yield a formula that is of exponential length.
We can use recursion to shorten the formula, but keep its meaning. M accepts a word w **if and only if** there is a path of length at most t from c_0 to c_{acc}, where t is the runtime of M, and is exponential in $s(n)$. This happens **if and only if** there is a path from c_0 to some c_m of length $\frac{t}{2}$, and a path from c_m to c_{acc} of length $\frac{t}{2}$ (we can assume w.l.o.g. that t is a power of 2).
Second attempt: Behold, another wrong attempt, but is more in the correct direction: Let us construct, inductively, a formula $\varphi_t(c_0, c_{\text{acc}})$ which states that c_{acc} is reachable from c_0 within at most t steps. More generally, we construct the formula $\varphi_k(c_0, c_{\text{acc}})$ which states that c_2 is reachable from c_1 within at most k steps. The formula is constructed as follows. First, if $k = 1$, then $\varphi_1(c_1, c_2)$ is simply the formula $\varphi(c_1, c_2)$ that we constructed earlier, stating that c_2 is consecutive to c_1. Then, we define:

$$\varphi_k(c_1, c_2) = \exists c_m \left(\varphi_{\frac{k}{2}}(c_1, c_m) \wedge \varphi_{\frac{k}{2}}(c_m, c_2) \right)$$

A truly brilliant solution, if only it was correct. The formula simply asks if there exists a configuration that functions as the middle configuration of the run. Clearly $\eta = \varphi_t(c_0, c_{\text{acc}})$ is true **if and only if** M accepts w. Also, since t is single exponential in $s(n)$ (that is $t = 2^{O(s(n))}$, then the recursion depth in constructing the formula is polynomial.
Well, the problem is that while the recursion depth is polynomial, the construction tree of the formula has degree 2, which means that it's a binary tree of polynomial depth, so it has an exponential number of nodes. That is, the formula is still too big. How can we overcome this?

Correct solution: Here comes a clever trick, which is the crux of the proof (are we not all so brilliantly bright?). Instead of asking whether there is a c_m that works with c_1 and with c_2, we combine these two cases into one, as follows:

$$\varphi_k(c_1, c_2) = \exists c_m \forall c_3, c_4 \left(((c_3 = c_1) \wedge (c_4 = c_m)) \vee ((c_3 = c_m) \wedge (c_4 = c_2))) \implies \varphi_{\frac{k}{2}}(c_3, c_4) \right.$$

Now the degree of the tree is 1, and it is still of polynomial depth, so the length of φ is polynomial, and so we are done. □

Note that we required that a TQF would have all its quantifiers in the beginning (this is sometimes called **prenex normal form**). However, our construction is in not in such a form, because of the condition on c_3 and c_4. This, however, is not a concern, because we can push the quantifiers out using the rules $\alpha \implies \exists x \beta \equiv \exists x (\alpha \implies \beta)$ and $\alpha \implies \forall x \beta \equiv \forall x (\alpha \implies \beta)$