Space complexity

Evgenij Thorstensen

V18

Space: The final frontier

There are interesting problems where we know the space complexity rather than time.

How space consumption behaves is also interesting.

Finally, space and time relate in non-obvious ways.

Space complexity

$\mathsf{SPACE}(f(n))$ is the class of languages with a DTM decider with space complexity $O(f(n))$.

Space complexity: Worst-case space usage $s_M(n)$, same as for time.

Can also define $\mathsf{NSPACE}(f(n))$, the class of languages with an NTM decider using $O(f(n))$ space.

Space is big

First, how powerful is space compared to time?

Space is big

First, how powerful is space compared to time?

Theorem

For every f, *we have* $\mathsf{NTIME}(f) \subseteq \mathsf{SPACE}(f)$.

We can simulate a time-bounded NTM with linear overhead.

If my NTM M is bounded by time $f(n)$, I use at most $f(n)$ tape cells on each branch.

The branch is given by at most $f(n)$ choices (transitions).

NTM simulation by space

To simulate a branch of the NTM, I preallocate $2f(n)$ cells.

Each pair of cells (x_i, y_i) will contain the transition choice and step number.

Beyond these I have my actual working tape.

Space simulation by time

What about the reverse? How many steps can a space-bounded decider possibly take?

Space simulation by time

What about the reverse? How many steps can a space-bounded decider possibly take?

Well, $f(n)$ tape cells give me $|\Sigma|^{f(n)}$ different tapes.

At each step, I must be in exactly one state, so $|\Sigma|^{f(n)} \times |Q|$ possible configurations.

For each configuration, I may be at any cell.

Theorem

For every $f(n) \geqslant n$, *we have* $\mathsf{SPACE}(f(n)) \subseteq \mathsf{TIME}(f(n) \cdot c^{f(n)})$ *for some* $c \in \mathbb{N}$.

Space simulation by time

What about the reverse? How many steps can a space-bounded decider possibly take?

Well, $f(n)$ tape cells give me $|\Sigma|^{f(n)}$ different tapes.

At each step, I must be in exactly one state, so $|\Sigma|^{f(n)} \times |Q|$ possible configurations.

For each configuration, I may be at any cell.

Theorem

For every $f(n) \geqslant n$, *we have* $\mathsf{SPACE}(f(n)) \subseteq \mathsf{TIME}(f(n) \cdot c^{f(n)})$ *for some* $c \in \mathbb{N}$.

If the machine runs for longer, it loops forever. Why?

Two interesting classes

$$\mathsf{PSPACE} = \bigcup_{k \in \mathbb{N}} \mathsf{SPACE}(n^k)$$

$$\mathsf{NPSPACE} = \bigcup_{k \in \mathbb{N}} \mathsf{NSPACE}(n^k)$$

Unlike for time, we also have the interesting classes

$$\mathsf{L} = \mathsf{SPACE}(\log n)$$

and

$$\mathsf{NL} = \mathsf{NSPACE}(\log n)$$

Some inclusions

$$\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} = \mathsf{NPSPACE} \subseteq \mathsf{EXP}$$

Exponential jumps from space to time, linear other way around.

First, PSPACE

First, we are going to do what cannot be done for P and NP, and prove that PSPACE = NPSPACE.

Theorem (Savitch)

For every $f(n) \geqslant n$, $\mathsf{NSPACE}(f(n)) \subseteq \mathsf{SPACE}(f(n)^2)$.

In other words, space-bounded NTMs can be simulated by DTMs with polynomial overhead.

Savitch, observations

Naive approach (like in the proof of $\mathsf{NTIME}(f) \subseteq \mathsf{SPACE}(f)$) won't work.

An NTM with $f(n)$ cells can take $f(n) \times c^{f(n)}$ steps. At each step I have a choice.

I need to avoid writing down these exponentially many choices.

Idea: Recursive binary search. If I recurse on the time bound of the NTM, I get $\log 2^{cf(n)} = cf(n)$ recursive calls.

The recursive search for acceptance

We will define a procedure $\mathit{CanYield}(c_1, c_2, t) \to \{0, 1\}$ that takes configurations c_1 and c_2 as input as well as a time bound t.

We will binary-search through the *choices* leading between configurations, looking for an accepting branch.

This will save us an exponential amount of space.

CanYield

CanYield(c_1, c_2, t):

1. If $t = 0$, test whether $c_1 = c_2$;
2. If $t > 0$, then loop through each configuration c_m:
   1. Run CanYield(c_1, c_m, $\frac{t}{2}$)
   2. Run CanYield(c_m, c_2, $\frac{t}{2}$)
   3. If both accept, accept.
3. If done with the loop, reject.

We will modify our NTM to have a clear accept *configuration*. We know that our NTM is time-bounded by $f(n) \times c^{f(n)}$. We will run

CanYield($c_{start}, c_{accept}, 2^{cf(n)}$). The depth of the recursion is $\log 2^{cf(n)} = cf(n)$.

Complexity analysis

Depth of recursion $cf(n)$.

At each call, store a new configuration c_m. Reuse this space when the recursion returns to try next configuration.

Total $O(f(n) \times cf(n)) = O(f(n)^2)$.

Observe that Savitch does not give us $\mathsf{L} = \mathsf{NL}$, since $\mathsf{SPACE}(\log n) \neq \mathsf{SPACE}(\log(n)^2)$.

PSPACE-completeness

Completeness if defined as before, given a notion of reduction $\leqslant_X$.

Polynomial space reductions bad, since $\mathsf{NPSPACE} = \mathsf{PSPACE}$.

We will stick to polynomial time reductions, $\leqslant_P$. A problem is complete for PSPACE is it is in PSPACE and every other problem there reduces to it.

Such problems exist, but are a bit exotic.

Generalizing SAT

In SAT, we ask for an assignment. Let's generalize this to asking questions about multiple assignments.

$\forall x(x \wedge y \to z)$ means "for every assignment to x, does there exist a satisfying assignment for the formula?"

Is the formula satisfiable regardless of x?

$\exists x \phi$ is just ϕ, is there an assignment? Could have $\exists$ on every variable.

Can nest these to be explicit.

TQBF

A TQBF formula is a SAT formula preceded by a string of quantifiers, one for each variable.

$\forall x.\exists y.\forall z.\phi(x, y, z)$

Easiest to think of it as a first-order formula where $\wedge, \vee, \neg$ are relations interpreted as required, and the universe is $\{0, 1\}$.

Order matters: $\forall x.\exists y(x \vee y) \wedge (\bar{x} \vee \bar{y})$ is true, while $\exists y.\forall x(x \vee y) \wedge (\bar{x} \vee \bar{y})$ is false.

TQBF, membership

The problem is: Given a TQBF formula, is it true?

Recursive algorithm to solve: For $\exists x \phi$, recurse with an or on the value of x, for $\forall x \phi$, recurse with an and.

TQBF, membership

The problem is: Given a TQBF formula, is it true?

Recursive algorithm to solve: For $\exists x \phi$, recurse with an or on the value of x, for $\forall x \phi$, recurse with an and.

For SAT, this recursion:

$\text{Solve}(\phi, i) = \text{Solve}(\phi[x_i = 1], i - 1) \vee \text{Solve}(\phi[x_i = 0], i - 1)$.

When out of variables, evaluate formula and return result.

For TQBF, same, but $\vee$ or $\wedge$ depends on the quantifier of x_i.

Analysis

Depth is number of variables, we store the values of the variables, space consumption $O(m)$, linear in the number of variables.

Therefore TQBF $\in$ PSPACE.

Why is this not in NP?