Space Complexity

Lecture 21

Binghui Peng

Space Complexity

For a TM, space complexity measures the number of tape cells used during a computation.

Definition:

$$\mathrm{SPACE}(f(n)) = \{L : L \text{ is decidable using } O(f(n)) \text{ tape cells}\}.$$

The Class PSPACE

Definition:

$$\mathrm{PSPACE} = \bigcup_{k \geq 1} \mathrm{SPACE}(n^k).$$

Time vs SPACE

A problem may need a huge amount of time but little space. For example, consider the following algorithm for SAT

- try all possible assignments one by one,
- keep only the current assignment in memory,
- reuse the same memory for the next assignment.

$\mathrm{NP} \subseteq \mathrm{PSPACE}$

Theorem

$\mathrm{NP} \subseteq \mathrm{PSPACE}$.

$\mathrm{NP} \subseteq \mathrm{PSPACE}$

Theorem

$$\mathrm{NP} \subseteq \mathrm{PSPACE}.$$

Proof Idea:

- A language is in NP if there exists a verifier M that could verify a certificate in polynomial time.
- The poly-space algorithm: Enumerate the certificate and accept if M accepts one of the certificate.

PSPACE-Complete

Definition: A language is **PSPACE-complete** if:

1. it belongs to PSPACE,
2. every language in PSPACE reduces to it in polynomial time.

PSPACE-Complete

Definition: A language is **PSPACE-complete** if:

1. it belongs to PSPACE,
2. every language in PSPACE reduces to it in polynomial time.

Question. Does PSPACE have a complete problem?

PSPACE-Complete

Definition: A language is **PSPACE-complete** if:

1. it belongs to PSPACE,
2. every language in PSPACE reduces to it in polynomial time.

Question. Does PSPACE have a complete problem?

Answer: Yes! TQBF is PSPACE-complete.

Quantified Boolean Formulas

A quantified Boolean formula allows arbitrary number of universal and existential quantifiers over Boolean variables. Example:

$$\exists x\ \forall y\ \exists z\ ((x \vee y) \wedge (\neg y \vee z)).$$

Semantics:

1. $\exists x$ means we may choose a value of x,
2. $\forall y$ means both values of y must work,
3. the quantifier-free part is then evaluated as an ordinary formula.

Examples

Example 1.

$$\exists x \, \forall y \, (x \vee y).$$

This is true: choose $x = 1$, and then $(x \vee y)$ is true for both values of y.

Example 2.

$$\forall x \, \exists y \, (x \wedge y).$$

This one is false, because when $x = 0$, no choice of y makes the formula true.

The Problem TQBF

Definition:

$$\mathrm{TQBF} = \{\Phi : \Phi \text{ is a true fully quantified Boolean formula}\}.$$

Commnent: SAT can be seen as special TQBF where all quantifiers are existential ($\exists$).

Theorem TQBF is PSACE-complete

Why TQBF $\in$ PSPACE

We can evaluate a quantified formula recursively. Algorithm idea:

1. If no quantifiers remain, evaluate the Boolean formula directly.
2. If the first quantifier is $\exists x$, recursively try $x = 0$ and $x = 1$, and OR the answers.
3. If the first quantifier is $\forall x$, recursively try $x = 0$ and $x = 1$, and AND the answers.

The recursion depth is the number of variables, so the algorithm uses only polynomial space.

Why TQBF Is PSPACE-Hard

The hard direction is deeper. Idea of the proof:

1. start with a polynomial-space TM M and input x,
2. consider the finite graph of configurations reachable using polynomial space,
3. use "divide and conquer" to build a quantified formula saying "there exists an accepting path through this configuration graph."

The formula uses recursive reachability, which keeps the size polynomial.

Comparing SAT and TQBF

Problem	Main quantifier pattern	Complexity class
SAT	exists assignment	NP
TQBF	alternating exists/for all	PSPACE

We don't know if $\mathrm{NP} \stackrel{?}{=} \mathrm{PSPACE}$ but most people believe $\mathrm{NP} \neq \mathrm{PSPACE}$.

We don't even know if $\mathrm{P} \stackrel{?}{=} \mathrm{PSPACE}$ but again, most people believe $\mathrm{P} \neq \mathrm{PSPACE}$.