15-251

Great Theoretical Ideas in Computer Science

Introduction to Computational Complexity III: Space Complexity and Circuit Complexity

February 10th, 2015

Today's Menu

Space complexity

Circuit complexity

Space Complexity

How should we define space complexity?

Should the input count?

Definition

A TM has space complexity $S(\cdot)$ if for every input *x*, it uses only $S(|x|)$ cells of the tape.

For 1-tape TM, $S(n) \geq n$ if the machine has to read the whole input.

So we actually consider a 2-tape TM:

- Tape 1 contains the input and is read-only
- Tape 2 is called the work tape, it is readable and writable.

The space complexity of the machine is defined with respect to the number of work tape cells it uses.

Example

$$L = \{0^k 1^k : k \geq 0\}$$

On input string w:

- Scan the input and **reject** if a 0 is found to the right of a 1.
- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape, cross off a single 0 and a single 1.
- If 0s remain but no 1s remain **or**
  1s remain but no 0s remain **reject**
- Else **accept**

(will need to copy the input to the work tape)

$O(n)$ space

Example

$$L = \{0^k 1^k : k \geq 0\}$$

On input string w:

- Scan the input and **reject** if a 0 is found to the right of a 1.
- Scan the input and count the number of 0s and 1s.
- If the counts are not the same **reject**
- Else **accept**

$O(\log n)$ **space**

Example

$$L = \{0^k 1^k : k \geq 0\}$$

```
def twoFingers(s):
    lo = 0
    hi = len(s)-1
    while (lo < hi):
        if (s[lo] != 0 or s[hi] != 1):
            return False
        lo += 1
        hi -= 1
    return True
```

$O(\log n)$ **space**

What can a log-space machine do?

Keep a pointer to a constant number of positions in the input.

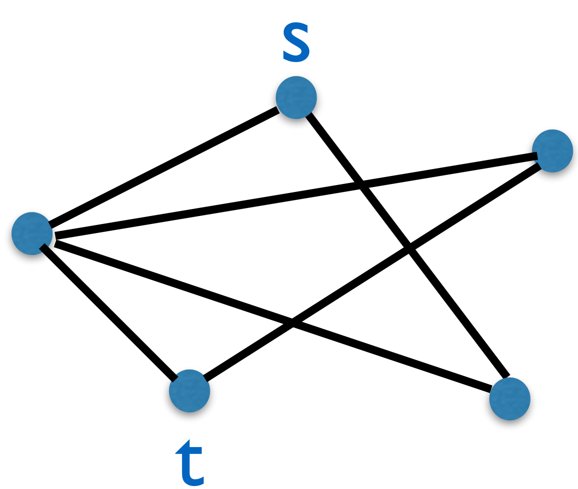
Count up to poly(n). $\log_2 n^k = k \log_2 n$

Keep logarithmic number of boolean variables.

Reachability problem

Input: A set of "cities", a set of "roads" between cities, and two specific cities s and t.

Output: Yes if we can reach t from s. No otherwise.



A "graph" with 5 nodes/vertices and 6 edges.

(imagine there are millions of vertices)

Omer Reingold (2004):

This problem is decidable using $O(\log n)$ space.

Example

Satisfiability (SAT)

Given a Boolean formula, is it satisfiable?

$$(x_1 \vee x_2) \wedge (x_3 \vee \neg x_2) \wedge \neg x_1$$

$$\exists x_1 \exists x_2 \ldots \exists x_n \ \varphi(x_1, x_2, \ldots, x_n)$$

QSAT (TQBF)

Given a quantified Boolean formula, is it true?

$$Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \ \varphi(x_1, x_2, \ldots, x_n)$$

(Each Q_i is $\exists$ or $\forall$. φ is allowed to have constants.)

Example

$$\Psi = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \; \varphi(x_1, x_2, \ldots, x_n)$$

QSAT can be decided using polynomial space.

size of φ: m

size of input: $O(n+m)$

Let $\Psi_{|x_1=b}$ be Ψ with Q_1 dropped,
and all occurrences of x_1 is replaced with b.

$A(\Psi)$:

if $n = 0$: ... just do it $\quad O(m)$ space

if $Q_1 = \exists$: output $(A(\Psi_{|x_1=0})$ or $A(\Psi_{|x_1=1}))$

if $Q_1 = \forall$: output $(A(\Psi_{|x_1=0})$ and $A(\Psi_{|x_1=1}))$

Example

$A(\Psi)$:

if $n = 0$: ... just do it $O(m)$ space

if $Q_1 = \exists$: output $(A(\Psi_{|x_1=0})$ or $A(\Psi_{|x_1=1}))$

if $Q_1 = \forall$: output $(A(\Psi_{|x_1=0})$ and $A(\Psi_{|x_1=1}))$

Let $S(n, m)$ = space used by algorithm *A*.

Observation: recursive calls $A(\Psi_{|x_1=0})$ and $A(\Psi_{|x_1=1})$ can use the same space.

$$S(n, m) = S(n-1, m) + O(n+m)$$
$$S(n, m) = O(n \cdot m + n^2)$$

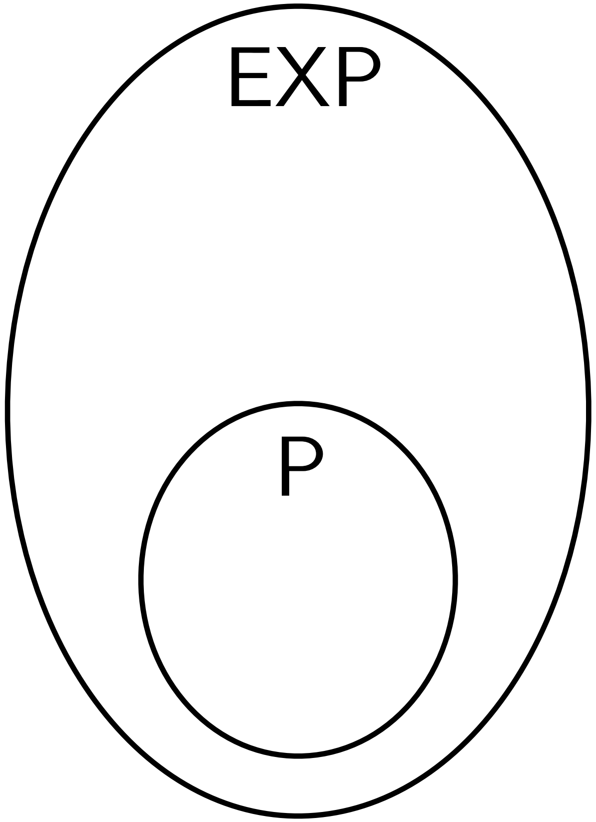
(at most quadratic in the input length)

Recall: time complexity classes

$$\mathrm{DTIME}(T(n)) = \{L : L \text{ is decided by an } O(T(n)) \text{ time algorithm.}\}$$

$$\mathsf{P} = \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(n^k)$$

$$\mathsf{EXP} = \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(2^{n^k})$$



$\mathsf{P} \subsetneq \mathsf{EXP}$

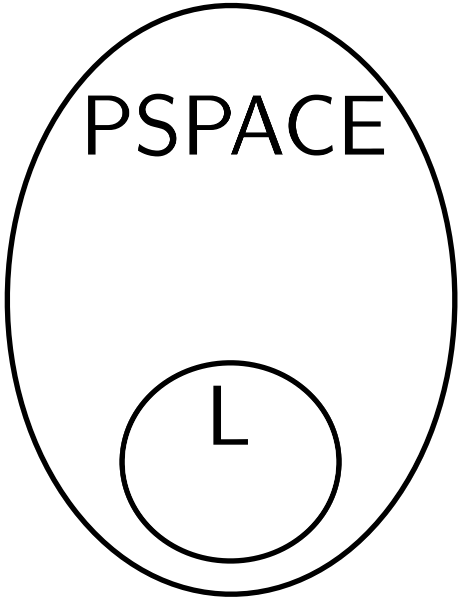
(Time hierarchy theorem)

Space complexity classes

$$\mathrm{DSPACE}(S(n)) = \{L : L \text{ is decided by an } O(S(n)) \text{ space algorithm.}\}$$

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

$$\mathsf{L} = \mathrm{DSPACE}(\log n)$$



$$\mathsf{L} \subseteq \mathsf{PSPACE}$$

$$\mathsf{L} = \mathsf{PSPACE} \ \ \textbf{???}$$

Hierarchy Theorems

Time Hierarchy Theorem:

Let $T(n)$ be a time-constructible function, and $\epsilon > 0.$

Then there is a problem which <u>cannot</u> be decided in time $T(n)$, but <u>can</u> be decided in time $T(n)^{1+\epsilon}$.

i.e., $\mathrm{DTIME}(T(n)) \subsetneq \mathrm{DTIME}(T(n)^{1+\epsilon})$

Space Hierarchy Theorem:

Let $S(n)$ be a space-constructible function, and $\epsilon > 0.$

Then there is a problem which <u>cannot</u> be decided in space $S(n)$, but <u>can</u> be decided in space $S(n)^{1+\epsilon}$.

i.e., $\mathrm{DSPACE}(S(n)) \subsetneq \mathrm{DSPACE}(S(n)^{1+\epsilon})$

Hierarchy Theorems

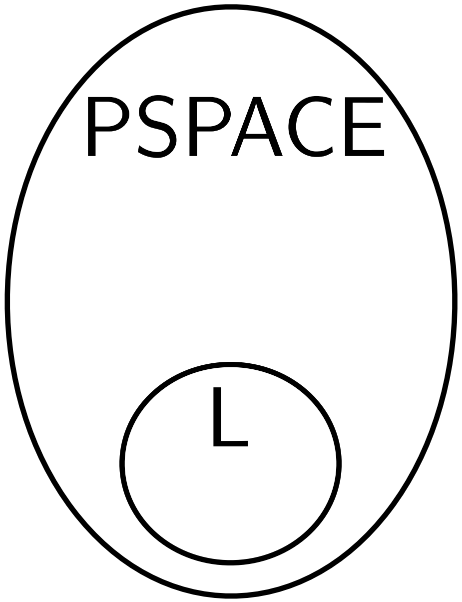
Space Hierarchy Theorem:

Let $S(n)$ be a space-constructible function, and $\epsilon > 0.$

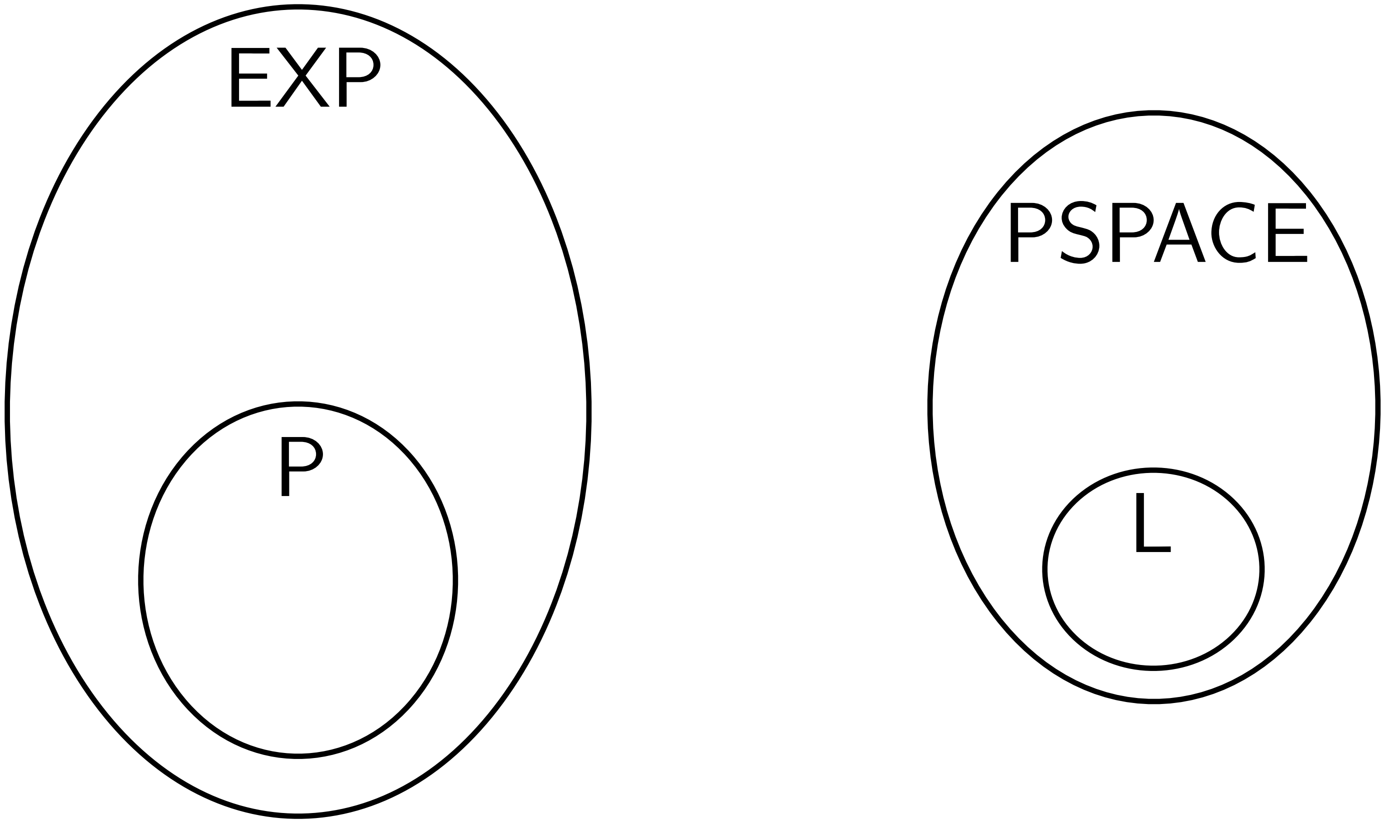
Then there is a problem which cannot be decided in space $S(n)$, but can be decided in space $S(n)^{1+\epsilon}$.

i.e., $\mathrm{DSPACE}(S(n)) \subsetneq \mathrm{DSPACE}(S(n)^{1+\epsilon})$

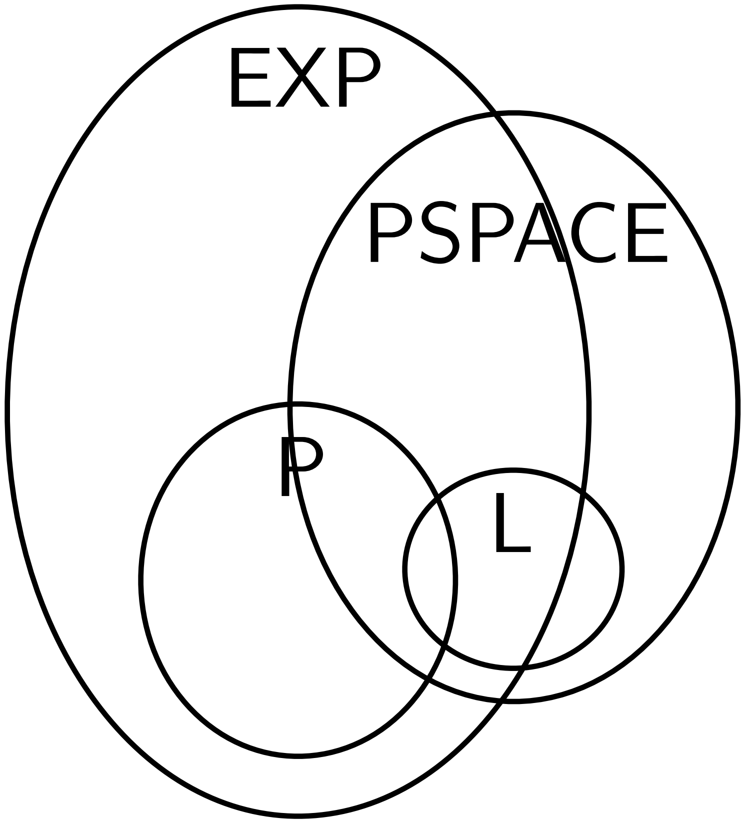
Corollary: L $\subsetneq$ PSPACE



Relationship between space and time



Relationship between space and time



???

Relationship between space and time

Theorem:

If a TM decides a language using $S(\cdot)$ space,
where $S(n) \geq \log_2 n$,
then it decides the language using $2^{O(S(n))}$ time.

Proof:

Recall a configuration of a TM is a string

$$uqv \qquad\qquad u, v \in \Gamma^*, \quad q \in Q$$

This is a snapshot of the TM's computation.

The information encoded in a configuration:

- current state
- the position of the tape head (work and input)
- contents of the tape (work)

Relationship between space and time

Proof (continued):

The information encoded in a configuration:

- current state
- the position of the tape head (work and input)
- contents of the tape (work)

If the TM takes t steps on a certain input,
there is a sequence of configurations: $c_1, c_2, \ldots, c_t$

Observation1: $c_i \neq c_j, \text{ for } i \neq j$

(otherwise the TM would be in an infinite loop.)

Observation2: $t \leq$ # possible configurations

Relationship between space and time

Proof (continued):

Observation1: $c_i \neq c_j, \;\text{for } i \neq j$

Observation2: $t \leq$ # possible configurations

Number of possible configurations is:

$$|Q| \cdot n \cdot S(n) \cdot |\Gamma|^{S(n)}$$

$$= C \cdot 2^{\log_2 n} \cdot 2^{\log_2 S(n)} \cdot 2^{(\log_2 |\Gamma|) S(n)}$$

$$= C \cdot 2^{\log_2 n + \log_2 S(n) + O(S(n))}$$

$$= 2^{O(S(n))}$$

So: $t \leq 2^{O(S(n))}$ $\square$

Relationship between space and time

Theorem:

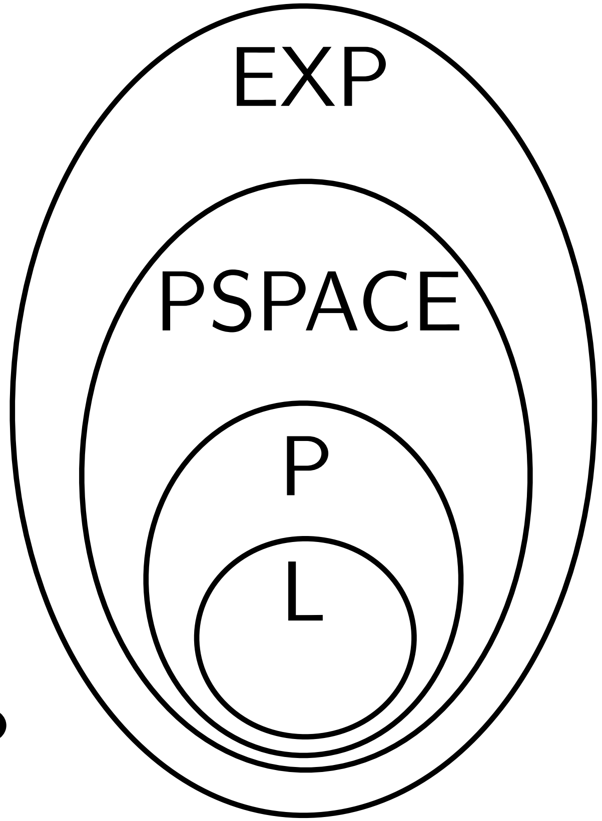
If a TM decides a language using $S(\cdot)$ space,
where $S(n) \geq \log_2 n$,
then it decides the language using $2^{O(S(n))}$ time.

Corollary 1: $\mathsf{L} \subseteq \mathsf{P}$ $\quad\left(2^{c \log_2 n} = n^c \right)$

Corollary 2: $\mathsf{PSPACE} \subseteq \mathsf{EXP}$

Relationship between space and time

$$\mathsf{L} \subseteq \mathsf{P} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP}$$



$$\mathsf{L} \subsetneq \mathsf{PSPACE} \quad \Longrightarrow$$

$$\mathsf{L} \subsetneq \mathsf{P} \quad \textbf{or} \quad \mathsf{P} \subsetneq \mathsf{PSPACE}$$

$$\mathsf{P} \subsetneq \mathsf{EXP} \quad \Longrightarrow$$

$$\mathsf{P} \subsetneq \mathsf{PSPACE} \quad \textbf{or} \quad \mathsf{PSPACE} \subsetneq \mathsf{EXP}$$

e.g., to show $\mathsf{P} \subsetneq \mathsf{PSPACE}$, **you need a language** A **:**

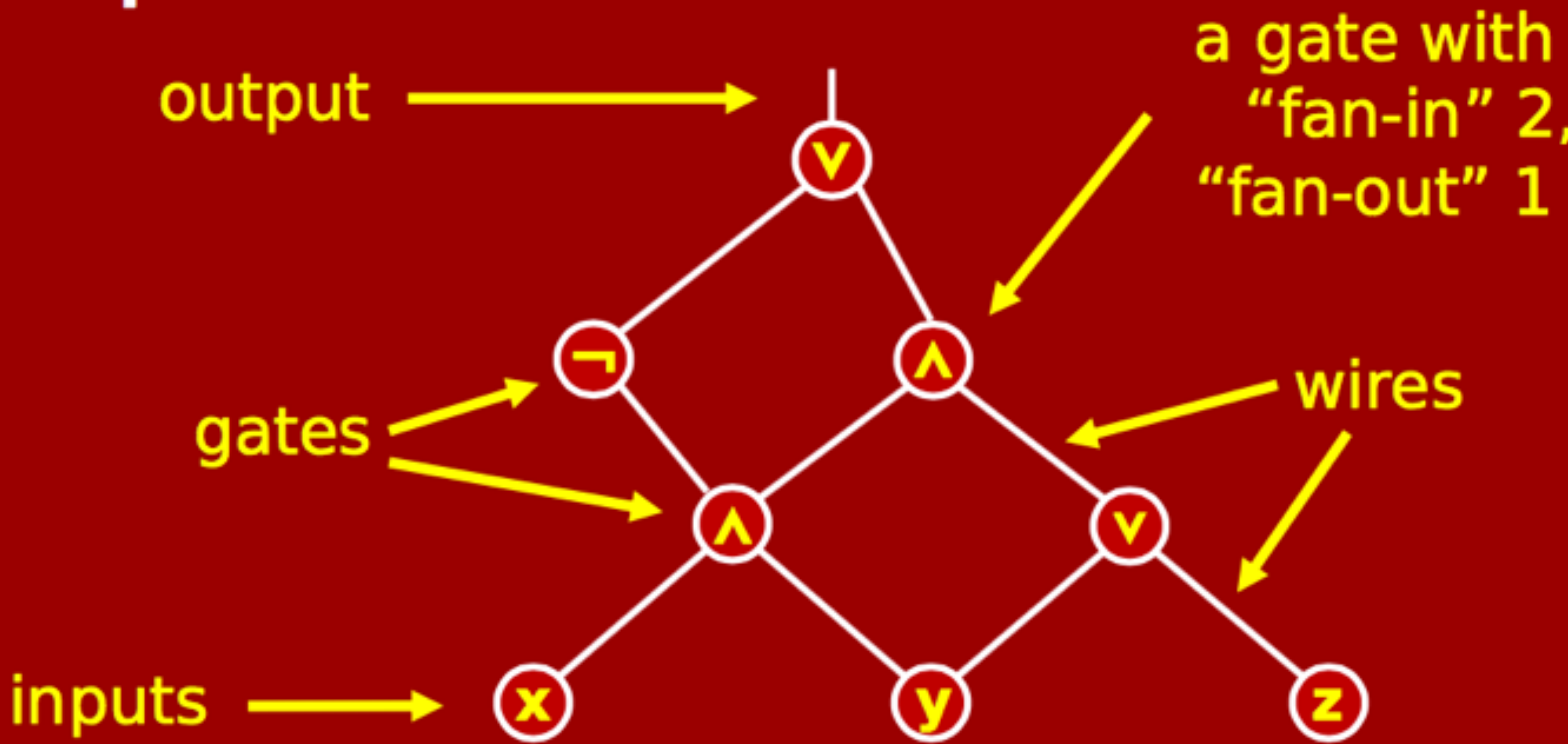
$$A \in \mathsf{PSPACE} \quad \textbf{but} \quad A \notin \mathsf{P}$$

Circuit Complexity

Recall the definition

$((x \wedge y) \wedge (y \vee z)) \vee \neg(x \wedge y)$ is a formula.

Depiction of the deduction:



Such a picture is called a Boolean **circuit**.

Recall the definition

A collection of gates and inputs connected by wires.

3 types of gates:

- binary AND gate
- binary OR gate
- unary NOT gate

Computes a function $f : \{0,1\}^n \to \{0,1\}$

(or decides a language $L_n \subseteq \{0,1\}^n$)

Important: A circuit can't handle all input lengths.
Need a circuit for each input length.

Circuit family

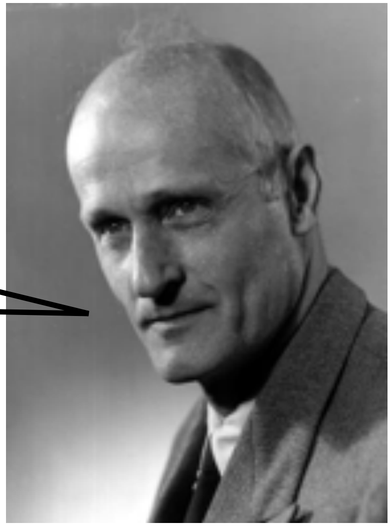
A circuit family C is a collection of circuits $(C_0, C_1, C_2, \ldots)$ where each C_n takes n input variables.

Let $L_n \subseteq \{0,1\}^n$ be the language decided by C_n .

Then $L = \bigcup_{n \in \mathbb{N}} L_n$ is the language decided by C .

Circuits vs TMs

An algorithm is a finite answer to infinite number of questions.



Stephen Kleene

A decider TM computes a function $f : \{0,1\}^* \rightarrow \{0,1\}$

A TM is has a constant size description.

Circuits vs TMs

A circuit family is an infinite answer to infinite number of questions.

Anil Ada

Perhaps not a very realistic model of computation.
(it is a "non-uniform" model)

Every function is computable!

It is still a very useful model to study!

Circuit size

The size of a circuit is the total number of gates (counting the input variables as gates too) in the circuit.

The size of a circuit family C is a function $s(\cdot)$ such that $s(n)$ is the size of C_n .

The circuit complexity of a language is the size of the minimal circuit family that decides the language.

(intrinsic complexity with respect to circuit size)

$L \in \mathrm{SIZE}(s(n))$ if there is a circuit family of size $O(s(n))$ that decides L.

Maximum circuit size for a function

Theorem:

For every language A, $A \in \mathrm{SIZE}(2^n)$.

Proof:

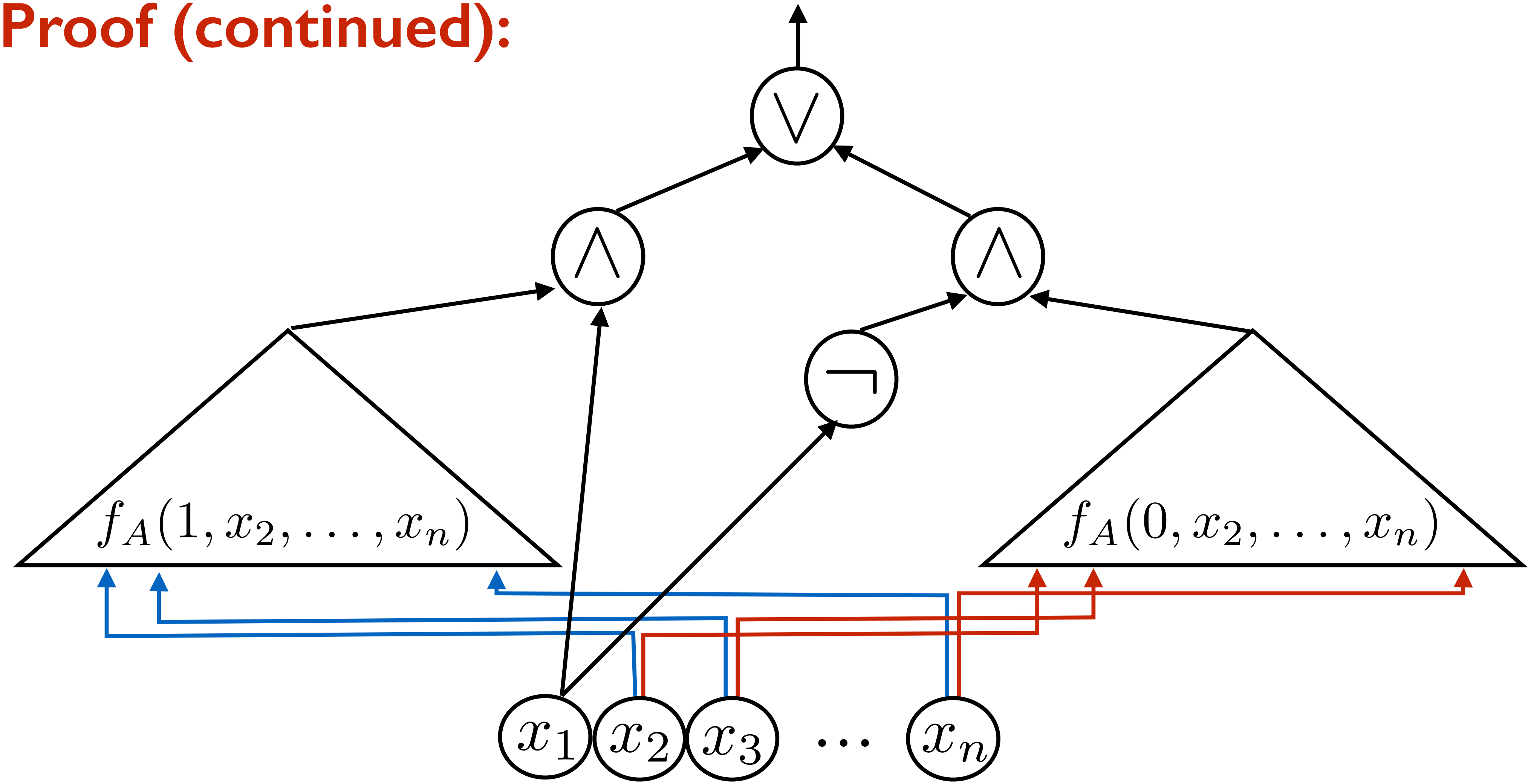
Let $f_A : \{0,1\}^n \to \{0,1\}$ correspond to A.

Observation:

$$f_A(x_1, x_2, \ldots, x_n) = (x_1 \wedge f_A(1, x_2, \ldots, x_n)) \vee (\neg x_1 \wedge f_A(0, x_2, \ldots, x_n))$$

Maximum size for a function

Proof (continued):



$$s(n) \leq 2s(n-1) + 5 \qquad\qquad s(1) \leq 3$$

$$\implies \; s(n) = O(2^n)$$

$\square$

Functions with exponential complexity

Theorem:

There is a language L whose circuit complexity is at least $2^n/4n$.

Proof:

Want to show: there is a function $f : \{0,1\}^n \to \{0,1\}$

that cannot be computed by a circuit of size $< 2^n/4n$.

Observation: # possible functions is 2^{2^n}

Claim1: # circuits of size s is $\leq 2^{4s \log s}$

Claim2: For $s \leq 2^n/4n$, $2^{4s \log s} < 2^{2^n}$

Then what we wanted to show follows immediately.

Functions with exponential complexity

Proof (continued):

Claim1: # circuits of size s is $\leq 2^{4s\log s}$

Claim2: For $s \leq 2^n/4n$, $2^{4s\log s} < 2^{2^n}$

Claim 2 is easy to verify. Proof of **Claim 1**:

For each circuit of size s,
we create a binary string of length $\leq 4s\log s$

This mapping will be injective, so Claim 1 will follow.

Number the gates: 1, 2, 3, 4, …, s

For each gate in the circuit, write down:

- type of the gate (2 bits)
- from which gates the inputs are coming from (2 log s bits)

Total: s(2 + 2 log s) bits $\square$

Remarks

That was due to Claude Shannon (1949).

Father of Information Theory.

Claude Shannon
(1916-2001)

A non-constructive argument.

In fact, it is easy to show that **most** functions require exponential size circuits.

Circuit complexity vs time complexity

Theorem:

If $A \in \mathrm{DTIME}(T(n))$, then $A \in \mathrm{SIZE}(T(n)^2)$.

i.e. $\mathrm{DTIME}(T(n)) \subseteq \mathrm{SIZE}(T(n)^2)$

Corollary:

If A cannot be computed by polynomial size circuits, then $A \notin$ P.

So to show P $\subsetneq$ PSPACE, find a language in PSPACE that cannot be computed by polynomial size circuits.

Current state of affairs:

After 60 years of research,
best lower bound for an explicit function: $5n - \text{peanuts}$

Advantages of working with circuits

A clean, simple mathematical definition.

Easy to create a hierarchy of problems.

- can restrict the depth (constant, log n, log^2 n, …)
- can allow other gates when the depth is restricted.
- can study monotone circuits.

Summary of
Introduction to Computational Complexity

Summary

Unlike computability, computational complexity depends on the computational model.

Definition of time complexity of an algorithm.

The CS way of comparing functions.
$O(\cdot), \Omega(\cdot), \Theta(\cdot)$

How you represent the input matters.
If the input is a number, imagine it has millions of digits.

Summary

Algorithms can do tricky things!
Always ask "Can we do better?"

Definition of the famous complexity class P.

Not all decidable problems can be efficiently decided.

Summary

Space complexity.

$L \subseteq P \subseteq PSPACE \subseteq EXP$

Circuit complexity.

- A nice and clean computational model.
- Related to time complexity.

We don't know how to prove lower bounds…