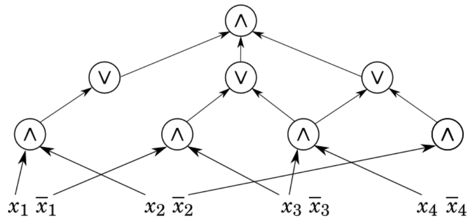
15-251
Great Ideas in Theoretical Computer Science

Lecture 15:
Boolean Circuits



March 6th, 2018

What is P ?

P

The theoretical divide between efficient and inefficient:

Why P ?

- Poly-time is **not** meant to mean "efficient in practice".
- Poly-time: extraordinarily better than brute force search.
- Poly-time: mathematical insight into problem's structure.
- Robust to notion of what is an
  elementary step, *what model we use*,
  reasonable encoding of input, *implementation details*.
- Nice closure property: Plug in a poly-time alg. into another poly-time alg. —> poly-time

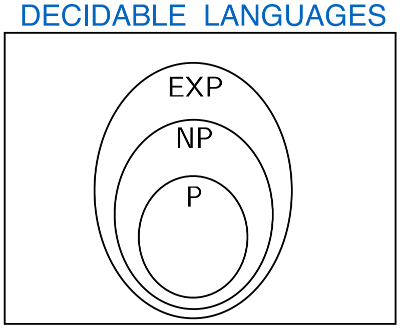
Why P ?

Summary:

What is NP ?

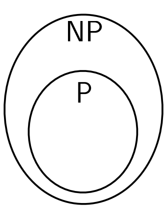
EXP

DECIDABLE LANGUAGES



NP :

What is NP ?



$\text{P} \stackrel{?}{=} \text{NP}$

asks whether these two sets are equal.

How would you show $\text{P} = \text{NP}$?

How would you show $\text{P} \neq \text{NP}$?

Boolean circuits are related to the P vs NP question in multiple ways.

Boolean Circuits

Some preliminary questions

What is a Boolean circuit?

- It is a computational model for computing decision problems (or computational problems).

We already have TMs. Why Boolean circuits?

-
-
-

- Circuits are good models to study *parallel computation*.
- Real computers are built with digital circuits.

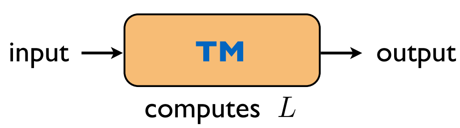
Dividing a problem according to length of input

$$\Sigma = \{0,1\}$$

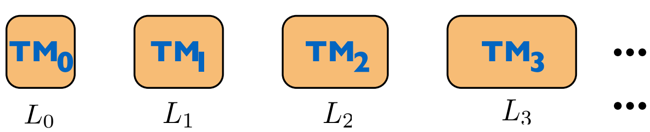
$L \subseteq \{0,1\}^*$	$f : \{0,1\}^* \to \{0,1\}$

Dividing a problem according to length of input

A TM is a finite object (finite number of states)
but can handle any input length.



Imagine a model where we allow the TM to change
with input length.



Dividing a problem according to length of input

So one machine does not compute L.

You use a family of machines:

$$(M_0, M_1, M_2, \ldots)$$

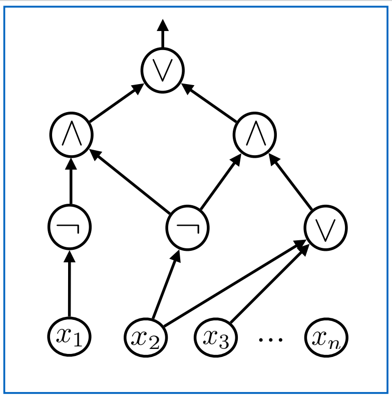
(Imagine having a different Python function for each input length.)

Is this a reasonable/realistic model of computation?!?

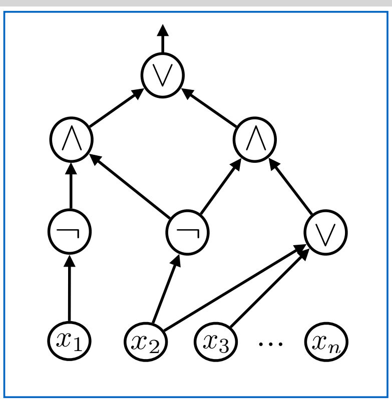
Boolean circuits work this way.
Need a separate circuit for each input length.
(but we still love them)

Boolean Circuit Definition

Picture of a circuit

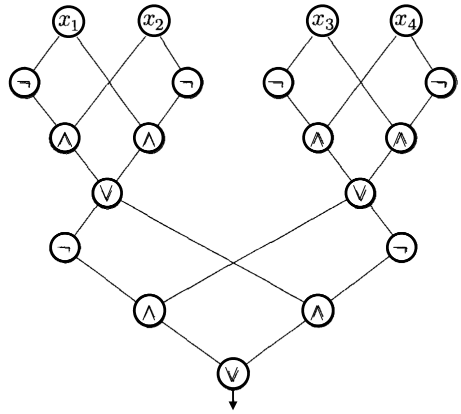


Picture of a circuit



Poll 1: What does this circuit compute ?

(sometimes circuits are drawn upside down)



How does a circuit decide a language?

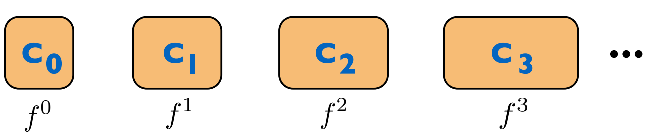
How do we measure the complexity of a circuit?

How can a circuit compute a language?

Given $f : \{0,1\}^* \to \{0,1\}$, write

$$f = (f^0, f^1, f^2, \ldots) \quad \text{where} \quad f^n : \{0,1\}^n \to \{0,1\}$$

Construct a circuit for each input length.



A circuit family C is

How can a circuit decide a language?

Circuit size and complexity

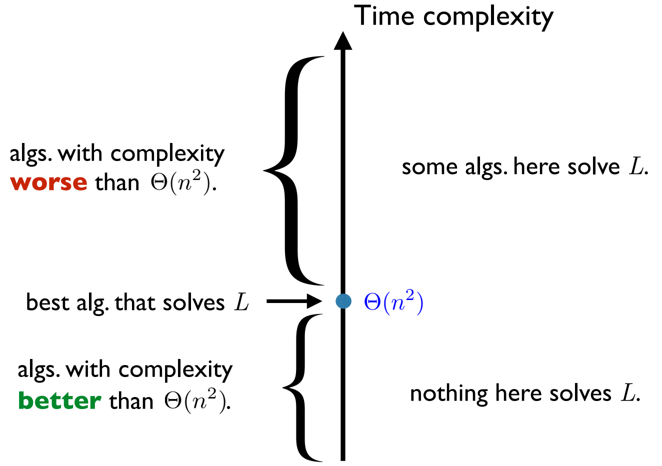
Definition (size of a circuit):

Definition (size of a circuit family):

Definition (circuit complexity):

Circuits vs TMs

Intrinsic complexity



Poll 2

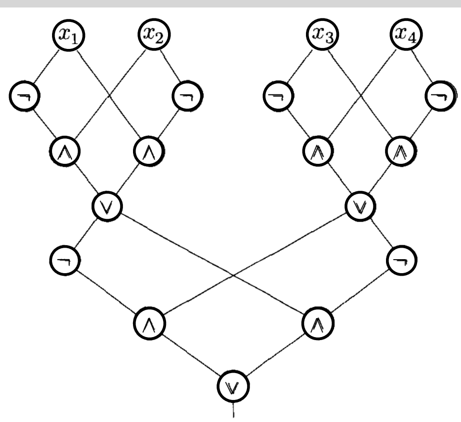
Let $f : \{0,1\}^* \to \{0,1\}$ be the parity decision problem.

$$f(x) = x_1 + \ldots + x_n \mod 2 \qquad (\text{where } n = |x|)$$

$$f(x) = x_1 \oplus \cdots \oplus x_n$$

What is the circuit complexity of this function?

Poll 2



The Big Picture Regarding Boolean Circuits

The big picture

Computability with respect to circuits

Theorem 1:

A universal exponential upper bound for all decision problems.

(We know this is not true in the TM model.)

The big picture

Limits of efficient computability with respect to circuits

Theorem 2 (Shannon's Theorem):

The big picture

Circuits can efficiently "simulate" TMs

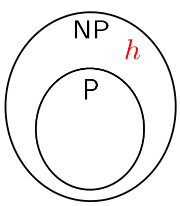
Theorem 3:

poly-time TM $\Longrightarrow$ poly-size circuits

Consequence of Theorem 3

poly-time TM $\Longrightarrow$ poly-size circuits

no poly-size circuits $\Longrightarrow$ no poly-time TM



To show P $\neq$ NP :

Find h in NP whose circuit complexity is more than any n^k.

Consequence of Theorem 3

So we can just work with circuits instead

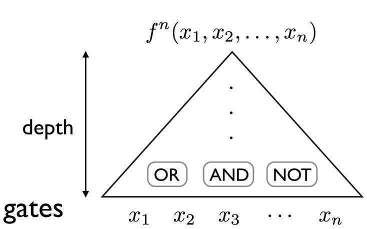
This is awesome in 2 ways:

1. Circuits: clean and simple definition of computation. "Just" a composition of AND, OR, NOT gates.

2. Restrict the circuit. Make it less powerful.

e.g. (i) restrict *depth*

(ii) restrict types of gates



Informal Poll

How many different functions $f : \{0,1\}^n \to \{0,1\}$ are there?

- n
- $2n$
- n^2
- 2^n
- 2^{2^n}
- none of the above
- beats me

Proof of Theorem 2

Theorem 2: Some functions are hard

Theorem: There exists a decision problem such that any circuit family computing it must have size at least $2^n/5n$.

Proof:

Theorem 2: Some functions are hard

Proof (continued):

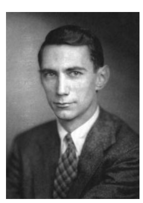
Theorem 2: Some functions are hard

Proof (continued):

Theorem 2: Some functions are hard

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 **almost all** functions require exponential size circuits.

Concluding Remarks

Boolean circuits: another model of computation.

(arguably simpler definition, easier to reason about)

no poly-size circuits $\Longrightarrow$ no poly-time TM

(can attack P vs NP problem with circuits)

CIRCUIT-SAT decision problem:

Given as input the description of a circuit,
output True if the circuit is "satisfiable".

Whether CIRCUIT-SAT is in P or not
is intimately related to the P vs NP question!