CS5371
Theory of Computation

Lecture 20: Complexity V
(Polynomial-Time Reducibility)

Objectives

- Polynomial Time Reducibility
- Prove Cook-Levin Theorem

Polynomial Time Reducibility

- Previously, we have learnt mapping reducibility, so that if a problem *A* can be 'mapped' in finite steps into another problem B, we can conclude that

  if B is decidable, *A* is decidable, or

  if B is recognizable, *A* is recognizable

- Suppose that we restrict the mapping reducibility to be done in polynomial time. What can we conclude?

Polynomial Time Reducibility (2)

We define (in this slide + in next slide):

Definition: A function $f:\Sigma^* \rightarrow \Sigma^*$ is a polynomial time computable function if some polynomial time TM M exists that halts with just f(w) on its tape, when started with input w

In other words, it is a computable function where the corresponding TM runs in polynomial time

Polynomial Time Reducibility (3)

Definition: Language A is polynomial time mapping reducible, or simply polynomial time reducible, to language B, written as $A \leq_P B$, if a polynomial time computable function f exists, where for each w,

$$w \in A \Leftrightarrow f(w) \in B$$

The function f is called a polynomial time reduction of A to B

Definition of NP-Complete

Definition: A language B is NP-complete if

1. B is in NP, and
2. every A in NP is polynomial time reducible to B

What is so special about NP-complete?

Properties of NP-Complete

Naturally, a NP-complete language is the "most difficult" language in NP for us to decide, because if it can be decided in polynomial time, every language in NP can be decided in polynomial time

In other words, we have…

Theorem: Suppose a language B is NP-complete. B is in P if and only if P = NP

Cook-Levin Theorem

Recall that Cook-Levin Theorem is the following:

Theorem: SAT is P if and only if P = NP

To prove the above theorem, it is equivalent if we prove:

Theorem: SAT is NP-complete

Proof of Cook-Levin

- To prove SAT is NP-complete, we need to do two things:
  1. Show SAT is in NP
  2. Show every other language in NP is polynomial time reducible to SAT

Proof of 1: Simple (Can you give a DTM verifier proof? Can you give an NTM decider proof?)

Proof of Cook-Levin (2)

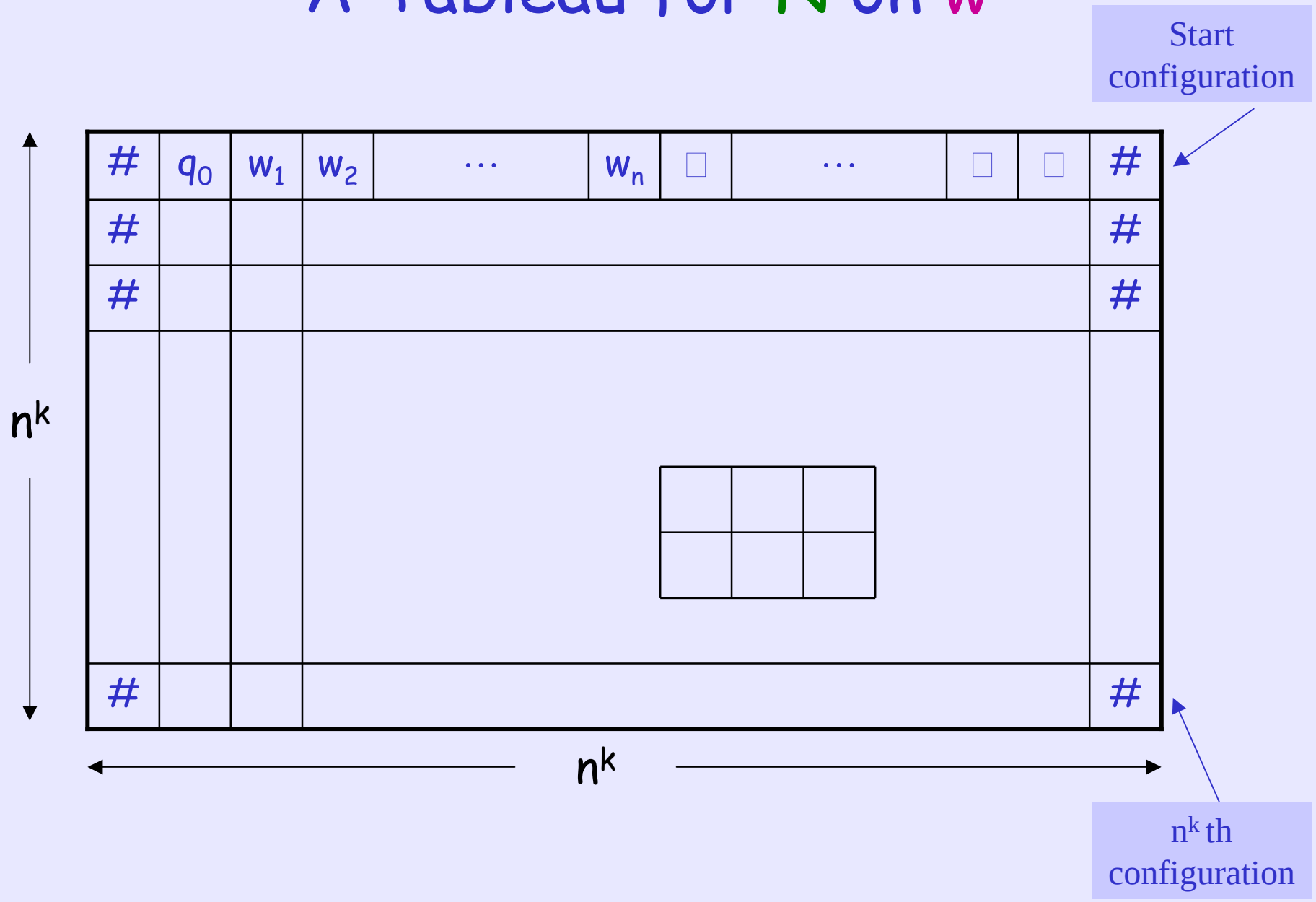
- Showing 2 is harder…
- Idea: We construct a polynomial time reduction for each A in NP to SAT.
- The reduction of A takes a string w and gives a Boolean formula F that simulates the NP machine N for A on input w
- If N accepts w, the corresponding satisfying assignment of F corresponds the accepting computation. Else, no satisfying assignment exists

Proof of Cook-Levin (3)

- Thus, we shall show that w is in A if and only if F is in SAT

Proof of 2: Let N be an NTM that decides A. Let the running time of N be n^k for some k. We define a tableau for N on an input string w to be an n^k by n^k table whose rows are the configurations of a branch of computation of N on w. For instance, (see next slide)

A Tableau for N on w



More on Tableau

- For convenience, we assume each configuration starts and ends with a # symbol
- The first row is the starting configuration, and each row follows from the previous row legally according to N's transition function
- A tableau is accepting if any row of the tableau is an accepting configuration

Proof of Cook-Levin (4)

- Note: every accepting tableau corresponds an accepting computation
- Thus, deciding whether N accepts w is equivalent to deciding whether an accepting tableau for N on w exists

- Our task now is to find the desired Boolean formula F based on N and w (what is the requirement of F?)

Proof of Cook-Levin (5)

- For each cell (i, j) in the tableau, and each **s** in $C = Q \cup \Gamma \cup \{\#\}$, we define a variable $x_{i,j,s}$. Intuitively, this variable is 1 means that cell (i,j) contains the symbol **s**
- Our formula F will correspond to a valid tableau, so we need to make sure when F is satisfiable:
  1. Each cell is occupied by exact 1 symbol
  2. The tableau has accepting configuration
  3. Each row is correct

Proof of Cook-Levin (5)

- In particular, we will use sub-formula to represent the above three cases, so that these sub-formula is satisfiable if the corresponding three cases are correct
- The final F is obtained by "And"-ing all these formula, so that if F is satisfiable, all three cases must be correct

Each Cell has only 1 symbol

- The following sub-formula ensures cell (i,j) contains at least one symbol:

$$f_{i,j,1} = \bigvee_{s \in C} x_{i,j,s}$$

- The following sub-formula ensures cell (i,j) contains at most one symbol:

$$f_{i,j,2} = \bigwedge_{s,t \in C,\, s \neq t} ((\neg x_{i,j,s}) \vee (\neg x_{i,j,t}))$$

Thus, $f_{i,j,1} \wedge f_{i,j,2}$ will ensure cell (i,j) has exactly one symbol, if F is satisfiable

Accepting Configuration

The following sub-formula ensures the tableau has an accepting configuration if F is satisfiable:

$$f_{accept} = \bigvee_{i,j} x_{i,j,q_{accept}}$$

Row is Legal

To ensure starting row is correct, we use the following sub-formula:

$$f_{start} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots \wedge$$
$$x_{1,n+2,w_n} \wedge x_{1,n+3,\square} \wedge \ldots \wedge x_{1,n^{k-1},\square} \wedge x_{1,n^k,\#}$$

To ensure the remaining rows are correct, we first define the concept of a window and legal window inside the tableau: (next slide)

Row is Legal (2)

A window at (i,j) refers to the 2x3 cells of (i,j), (i,j+1), (i,j+2), (i+1,j), (i+1,j+1), and (i+1,j+2)

A legal window is a window that does not violate the actions specified by the N's transition function, considering the configuration of each row is following legally from the configuration in the row above

Row is Legal (3)

E.g.,

a	q_1	b
q_2	a	c

This window is legal if there is a transition $\delta(q_1,b) = (q_2,c,L)$

a	q_1	b
a	a	q_2

This window is legal if there is a transition $\delta(q_1,b) = (q_2,a,R)$

a	a	q_1
a	a	b

This window is legal if there is a transition $\delta(q_1,c) = (q_2,b,R)$ for some c and q_2

Row is Legal (4)

E.g.,

#	a	b
#	a	b

This window is also legal

a	b	a
a	b	q_2

This window is legal if there is a transition $\delta(q_1,b) = (q_2,c,L)$ for some q_1, b, and c

a	a	a
b	a	a

This window is legal if there is a transition $\delta(q_1,a) = (q_2,b,L)$ for some q_1 and q_2

Row is Legal (5)

E.g.,

a	b	b
a	a	b

a	q_1	b
q_2	a	q_2

a	q_1	a
q_2	c	b

All these windows cannot be legal, why?

Row is Legal (6)

Note the the window containing the state symbol in the center top cell guarantees that the corresponding three lower cells are updated consistently with the transition function

Thus, if the upper configuration is a legal, so is the lower configuration, and so is every configuration that follows below

Row is Legal (7)

Based on the legal window concept, the following sub-formula ensures that each row are following correctly:

$f_{move} = \bigwedge_{1 \le i,j \le n^k-2}$ (window at (i,j) is legal)

and the text "window at (i,j) is legal" can be replaced by:

$$\bigvee_{a1,a2,\ldots,a6 \text{ is a legal window}} (x_{i,j,a1} \wedge x_{i,j+1,a2} \wedge x_{i,j+2,a3} \wedge x_{i+1,j,a4} \wedge x_{i+1,j+1,a5} \wedge x_{i+1,j+2,a6})$$

Proof of Cook-Levin (6)

Thus, if

$$F = (\bigwedge_{i,j} (f_{i,j,1} \wedge f_{i,j,2})) \wedge f_{accept} \wedge f_{start} \wedge f_{move}$$

then F is satisfiable implies N has an accepting computation on input w ➔ N accepts w

Conversely, we can see that if N accepts w, there must be an accepting computation, and F has a satisfying assigment ➔ F is satisfiable

Proof of Cook-Levin (7)

The above construction of F gives a reduction from deciding a language in NP to deciding whether a formula is satisfying

To show SAT is NP-complete, it remains to show that the above construction is done in polynomial time (in terms of the length of the input w)

Proof of Cook-Levin (8)

Given w of length n,

- the sub-formula f_{start} can be constructed in $O(n^k)$ time
- the sub-formula $\wedge_{i,j} (f_{i,j,1} \wedge f_{i,j,2})$, f_{accept} and f_{move} can be constructed in $O(n^{2k})$ time [why??]

As any language in NP is polynomial time reducible to SAT and SAT is in NP ➔ SAT is NP-complete

Next Time

- More NP-complete problems