

CS151

Complexity Theory

Lecture 2

April 5, 2017

Interlude

- In an ideal world, given language L
 - state an algorithm deciding L
 - **prove** that no algorithm does better
- we are pretty good at part 1
- we are currently **completely helpless** when it comes to part 2, for most problems that we care about

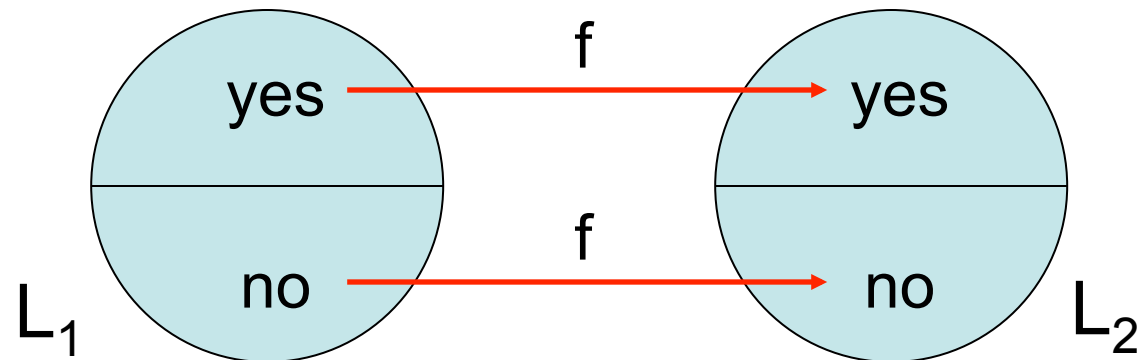
Interlude

- in place of part 2 we can
 - relate the difficulty of problems to each other via **reductions**
 - prove that a problem is a “hardest” problem in a complexity class via **completeness**
- powerful, successful surrogate for lower bounds

Reductions

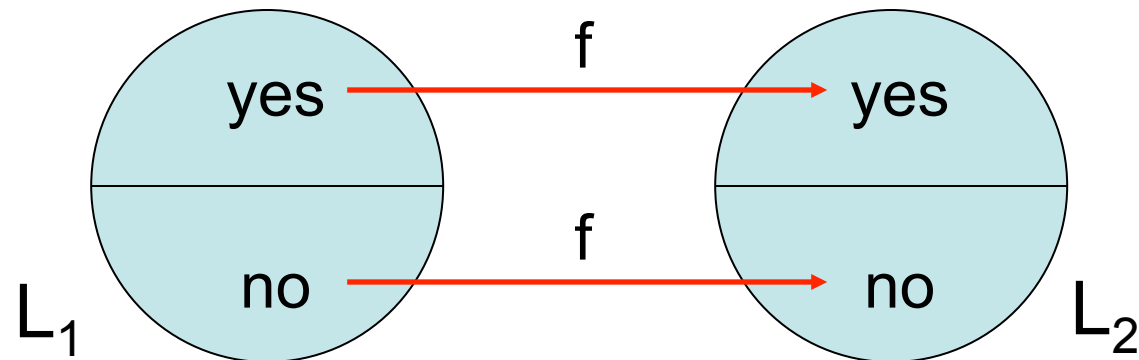
- **reductions** are the main tool for relating problems to each other
- given two languages L_1 and L_2 we say “ L_1 **reduces to** L_2 ” and we write “ $L_1 \leq L_2$ ” to mean:
 - there exists an efficient (for now, poly-time) algorithm that computes a function f s.t.
 - $x \in L_1$ implies $f(x) \in L_2$
 - $x \notin L_1$ implies $f(x) \notin L_2$

Reductions



- positive use: given new problem L_1 reduce it to L_2 that we know to be in \mathbf{P} . Conclude L_1 in \mathbf{P} (how?)
 - e.g. bipartite matching \leq max flow
 - formalizes “ L_1 as easy as L_2 ”

Reductions



- **negative use:** given new problem L_2 reduce L_1 (that we believe not to be in \mathbf{P}) to it. Conclude L_2 *not* in \mathbf{P} if L_1 *not* in \mathbf{P} (how?)
 - e.g. satisfiability \leq graph 3-coloring
 - formalizes “ L_2 as hard as L_1 ”

Reductions

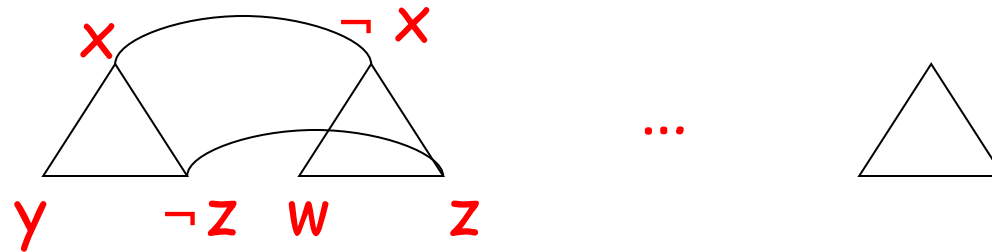
- Example reduction:
 - 3SAT = { φ : φ is a 3-CNF Boolean formula that has a satisfying assignment }
(3-CNF = AND of OR of ≤ 3 literals)
 - IS = { (G, k) | G is a graph with an independent set $V' \subseteq V$ of size $\geq k$ }
(ind. set = set of vertices no two of which are connected by an edge)

Ind. Set is NP-complete

The reduction f: given

$$\varphi = (x \vee y \vee \neg z) \wedge (\neg x \vee w \vee z) \wedge \dots \wedge (\dots)$$

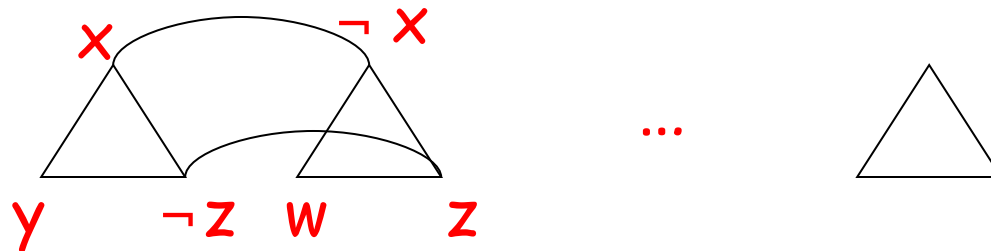
we produce graph G_φ :



- one triangle for each of m clauses
- edge between every pair of contradictory literals
- set $k = m$

Reductions

$$\varphi = (x \vee y \vee \neg z) \wedge (\neg x \vee w \vee z) \wedge \dots \wedge (\dots)$$



- Claim: φ has a satisfying assignment if and only if G has an independent set of size at least k
 - proof?
- Conclude that $3SAT \leq IS$.

Completeness

- complexity class **C**
- language L is **C-complete** if
 - L is in **C**
 - every language in **C** reduces to L
- very important concept
- formalizes “ L is hardest problem in complexity class **C**”

Completeness

- Completeness allows us to reason about the entire class by thinking about a single concrete problem
- related concept: language L is **C-hard** if
 - every language in **C** reduces to L

Completeness

- May ask: how to show *every* language in **C** reduces to L?
 - in practice, shown by reducing **known C-complete** problem to L
 - often not hard to find “1st” C-complete language, but it might not be “natural”

Completeness

– Example:

NP = the set of languages L where

$$L = \{ x : \exists y, |y| \leq |x|^k, (x, y) \in R \}$$

and R is a language in **P**.

one **NP**-complete language “bounded halting”:

$$\text{BH} = \{ (M, x, 1^m) : \exists y, |y| \leq |x|^k \text{ s.t. } M \\ \text{accepts } (x, y) \text{ in at most } m \text{ steps} \}$$

– challenge is to find **natural** complete problem

– Cook 71 : SAT **NP**-complete

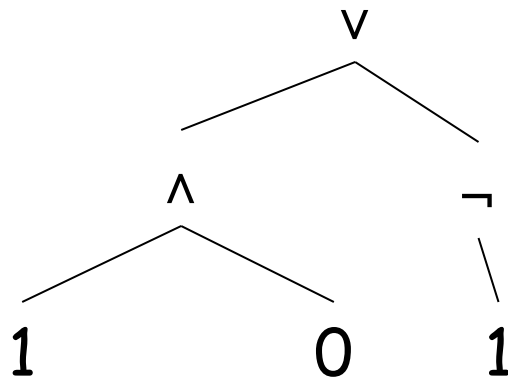
Summary

- problems
 - function, decision
 - language = set of strings
- complexity class = set of languages
- efficient computation identified with efficient computation on Turing Machine
 - single-tape, multi-tape
 - diagonalization technique: HALT undecidable
- TIME and SPACE classes
- reductions
- **C**-completeness, **C**-hardness

Time and Space

A motivating question:

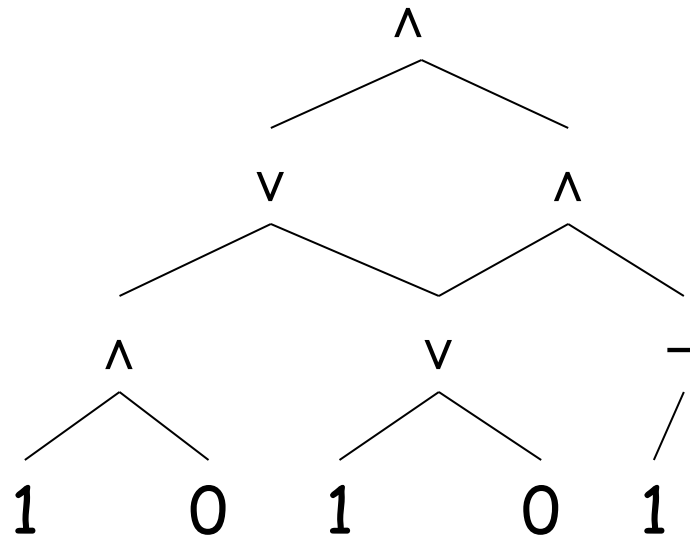
- Boolean formula with n nodes
- evaluate using $O(\log n)$ space?



- depth-first traversal requires storing intermediate values
- idea: short-circuit ANDs and ORs when possible

Time and Space

- Can we evaluate an n node Boolean **circuit** using $O(\log n)$ space?



Time and Space

- Recall:
 - **TIME(f(n)), SPACE(f(n))**
- Questions:
 - how are these classes related to each other?
 - how do we define **robust** time and space classes?
 - what problems are contained in these classes? complete for these classes?

Outline

- Why big-oh? Linear Speedup Theorem
- Hierarchy Theorems
- Robust Time and Space Classes
- Relationships between classes
- Some complete problems

Linear Speedup

Theorem: Suppose TM M decides language L in time $f(n)$. Then for any $\epsilon > 0$, there exists TM M' that decides L in time

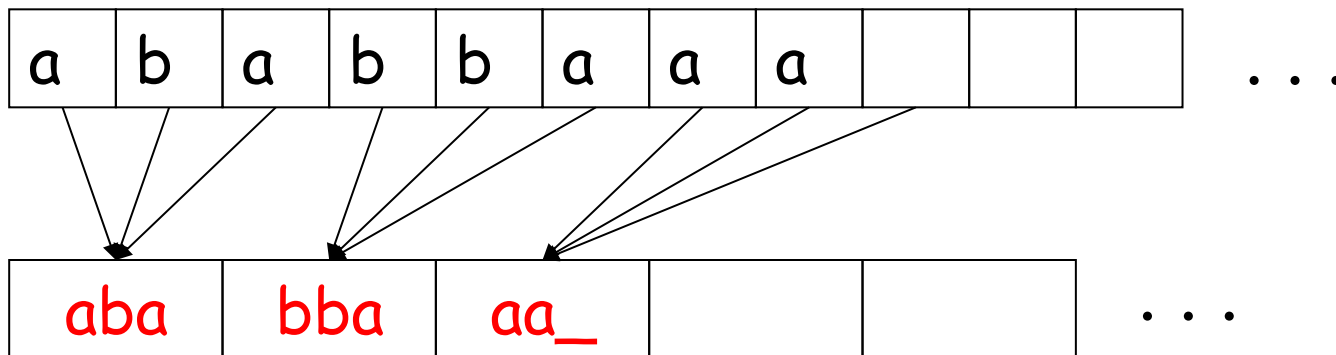
$$\epsilon f(n) + n + 2.$$

- Proof:
 - simple idea: increase “word length”
 - M' will have
 - one more tape than M
 - m -tuples of symbols of M
 - many more states

$$\Sigma_{\text{new}} = \Sigma_{\text{old}} \cup \Sigma_{\text{old}}^m$$

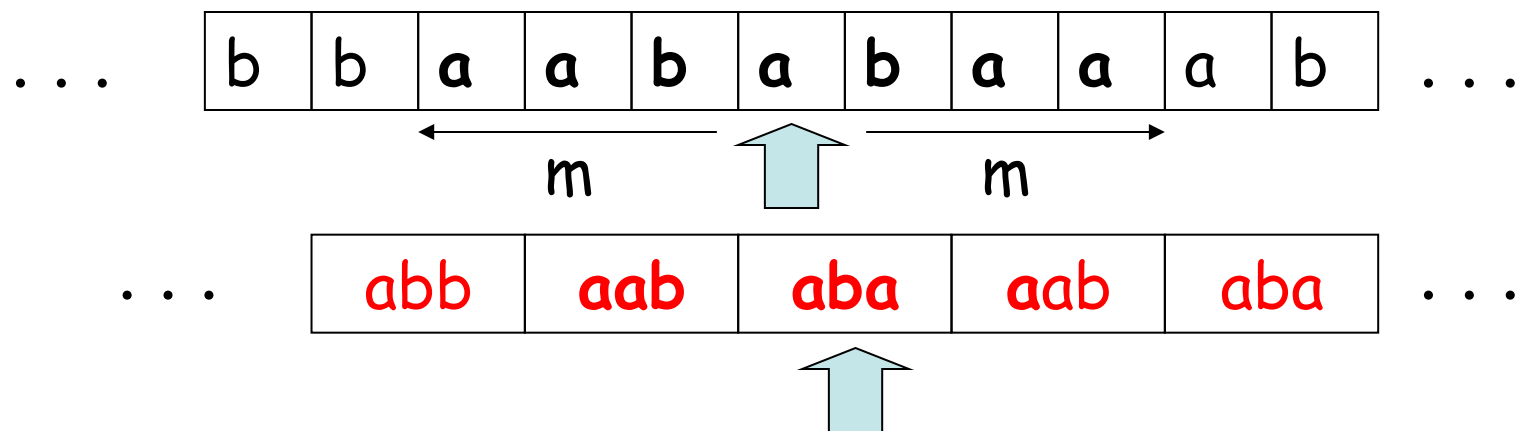
Linear Speedup

- part 1: compress input onto fresh tape



Linear Speedup

- part 2: simulate M , m steps at a time



- 4 (L,R,R,L) steps to read relevant symbols, “remember” in state
- 2 (L,R or R,L) to make M 's changes

Linear Speedup

- accounting:
 - part 1 (copying): $n + 2$ steps
 - part 2 (simulation): $6 (f(n)/m)$
 - set $m = 6/\epsilon$
 - total: $\epsilon f(n) + n + 2$

Theorem: Suppose TM M decides language L in space $f(n)$. Then for any $\epsilon > 0$, there exists TM M' that decides L in space $\epsilon f(n) + 2$.

- Proof: same.

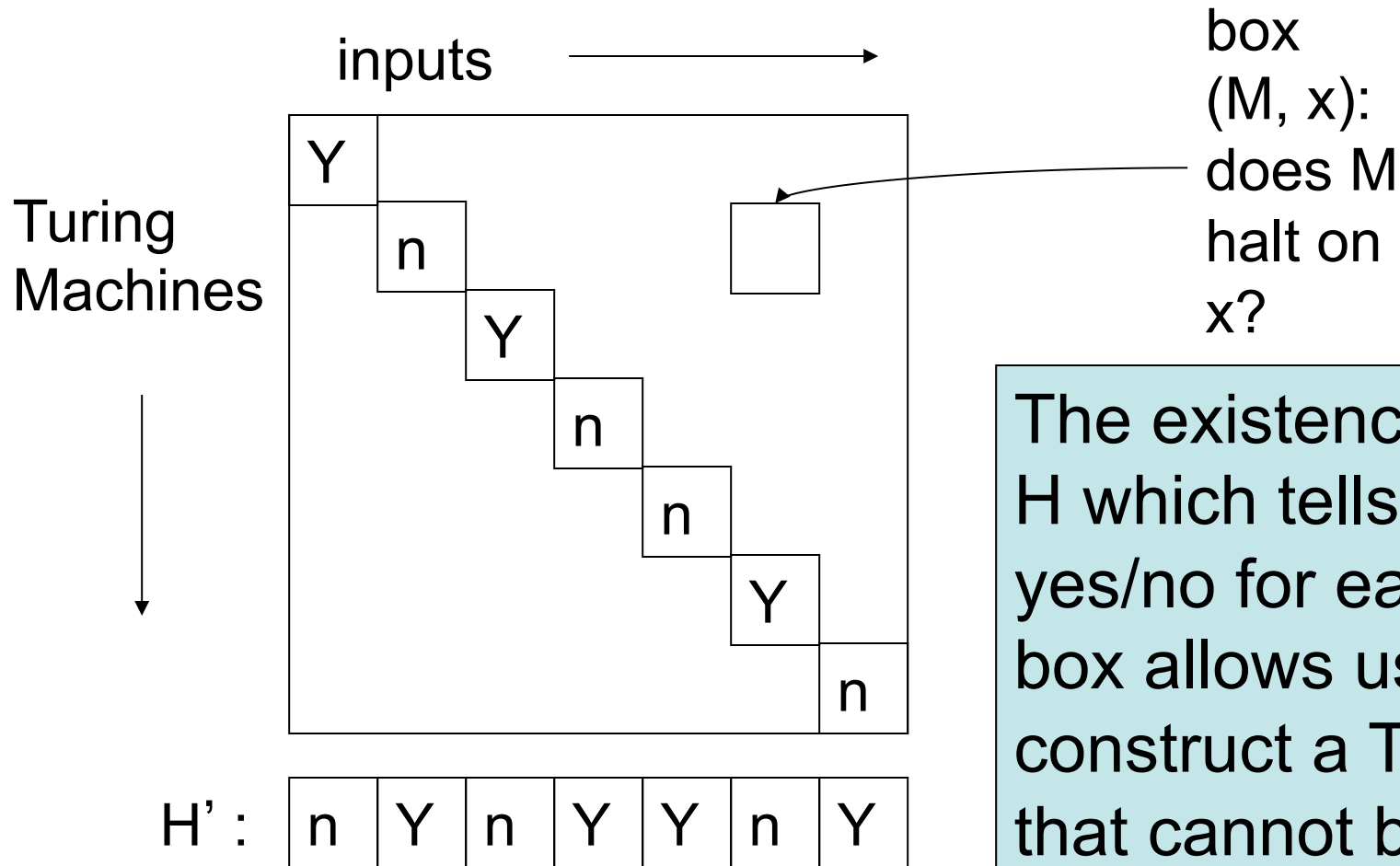
Time and Space

- Moral: big-oh notation **necessary** given our model of computation
 - Recall: $f(n) = O(g(n))$ if there exists c such that $f(n) \leq c g(n)$ for all sufficiently large n .
 - TM model incapable of making distinctions between time and space usage that differs by a constant.
- In general: interested in course distinctions not affected by model
 - e.g. simulation of k -string TM running in time $f(n)$ by single-string TM running in time $O(f(n)^2)$

Hierarchy Theorems

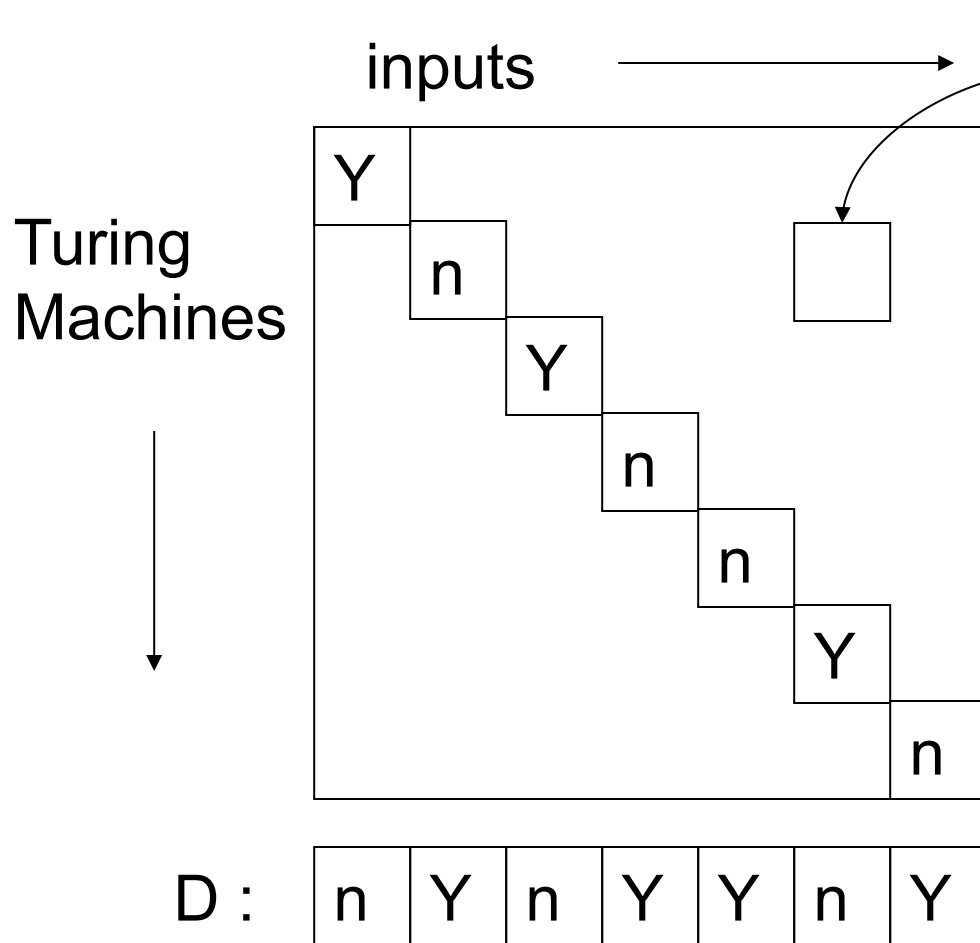
- Does **genuinely** more time permit us to decide new languages?
- how can we construct a language L that is *not* in **TIME**($f(n)$)...
 - idea: same as “HALT undecidable”
diagonalization and simulation

Recall proof for Halting Problem



The existence of H which tells us yes/no for each box allows us to construct a TM H' that cannot be in the table.

Time Hierarchy Theorem



box (M, x) : does M accept x in time $f(n)$?

- TM **SIM** tells us yes/no for each box in time $g(n)$
- rows include all of **TIME($f(n)$)**
- construct TM D running in time $g(2n)$ that is not in table

Time Hierarchy Theorem

Theorem (Time Hierarchy Theorem): For every *proper complexity function* $f(n) \geq n$:
 $\text{TIME}(f(n)) \subsetneq \text{TIME}(f(2n)^3)$.

- more on “proper complexity functions” later...

Proof of Time Hierarchy Theorem

- Proof:
 - SIM is TM deciding language
 $\{ \langle M, x \rangle : M \text{ accepts } x \text{ in } \leq f(|x|) \text{ steps} \}$
 - Claim: SIM runs in time $g(n) = f(n)^3$.
 - define new TM D: on input $\langle M \rangle$
 - if SIM accepts $\langle M, M \rangle$, reject
 - if SIM rejects $\langle M, M \rangle$, accept
 - D runs in time $g(2n)$

Proof of Time Hierarchy Theorem

- Proof (continued):
 - suppose M in **TIME**($f(n)$) decides $L(D)$
 - $M(\langle M \rangle) = \text{SIM}(\langle M, M \rangle) \neq D(\langle M \rangle)$
 - but $M(\langle M \rangle) = D(\langle M \rangle)$
 - contradiction.

Proof of Time Hierarchy Theorem

- Claim: there is a TM SIM that decides $\{\langle M, x \rangle : M \text{ accepts } x \text{ in } \leq f(|x|) \text{ steps}\}$ and runs in time $g(n) = f(n)^3$.
- Proof sketch: SIM has 4 work tapes
 - contents and “virtual head” positions for M’s tapes
 - M’s transition function and state
 - $f(|x|)$ “+”s used as a clock
 - scratch space

Proof of Time Hierarchy Theorem

- contents and “virtual head” positions for M 's tapes
 - M 's transition function and state
 - $f(|x|)$ “+”s used as a clock
 - scratch space
- initialize tapes
 - simulate step of M , advance head on tape 3; repeat.
 - can check running time is as claimed.
- Important detail: need to initialize tape 3 in time $O(f(n))$

Proper Complexity Functions

- Definition: f is a **proper complexity function** if
 - $f(n) \geq f(n-1)$ for all n
 - there exists a TM M that outputs exactly $f(n)$ symbols on input 1^n , and runs in time $O(f(n) + n)$ and space $O(f(n))$.

Proper Complexity Functions

- includes all reasonable functions we will work with
 - $\log n$, \sqrt{n} , n^2 , 2^n , $n!$, ...
 - if f and g are proper then $f + g$, fg , $f(g)$, f^g , 2^g are all proper
- can mostly ignore, but be aware it is a genuine concern:
- Theorem: \exists non-proper f such that **$\text{TIME}(f(n)) = \text{TIME}(2^{f(n)})$** .

Hierarchy Theorems

- Does **genuinely** more **space** permit us to decide new languages?

Theorem (Space Hierarchy Theorem): For every *proper complexity function* $f(n) \geq \log n$:

$$\mathbf{SPACE(f(n)) \subsetneq SPACE(f(n) \log f(n)).}$$

- Proof: same ideas.

Robust Time and Space Classes

- What is meant by “robust” class?
 - no formal definition
 - reasonable changes to model of computation shouldn't change class
 - should allow “modular composition” – calling subroutine in class (for classes closed under complement...)

Robust Time and Space Classes

- Robust time and space classes:

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

$$\mathbf{PSPACE = \bigcup_k SPACE(n^k)}$$

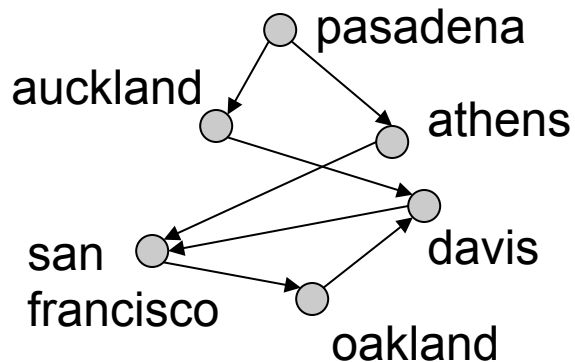
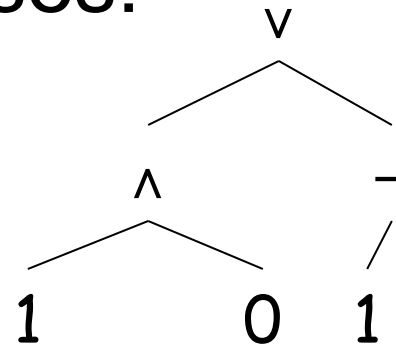
$$\mathbf{P = \bigcup_k TIME(n^k)}$$

$$\mathbf{EXP = \bigcup_k TIME(2^{n^k})}$$

Time and Space Classes

- Problems in these classes:

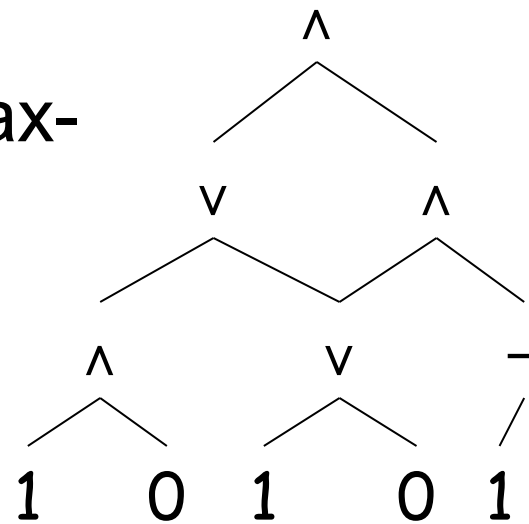
L : FVAL, integer multiplication, most reductions...



PSPACE : generalized geography, 2-person games...

Time and Space Classes

P : CVAL, linear programming, max-flow...



EXP : SAT, all of NP and much more...

Relationships between classes

- How are these four classes related to each other?

- Time Hierarchy Theorem implies

$$P \subsetneq EXP$$

$$- P \subseteq TIME(2^n) \subsetneq TIME(2^{(2^n)^3}) \subseteq EXP$$

- Space Hierarchy Theorem implies

$$L \subsetneq PSPACE$$

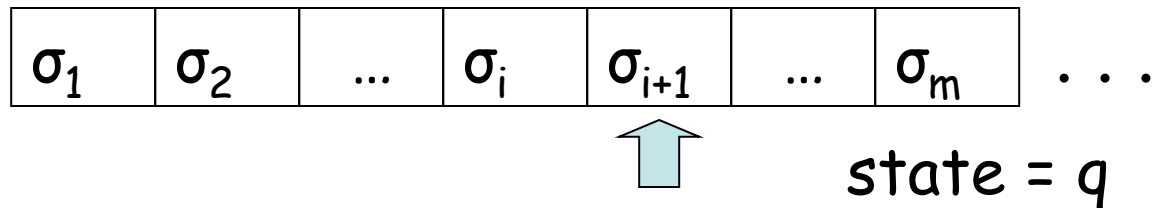
$$- L = SPACE(\log n) \subsetneq SPACE(\log^2 n) \subseteq PSPACE$$

Relationships between classes

- Easy: **$P \subseteq PSPACE$**
- L vs. P, PSPACE vs. EXP ?

Relationships between classes

- Useful convention: **Turing Machine configurations**. Any point in computation



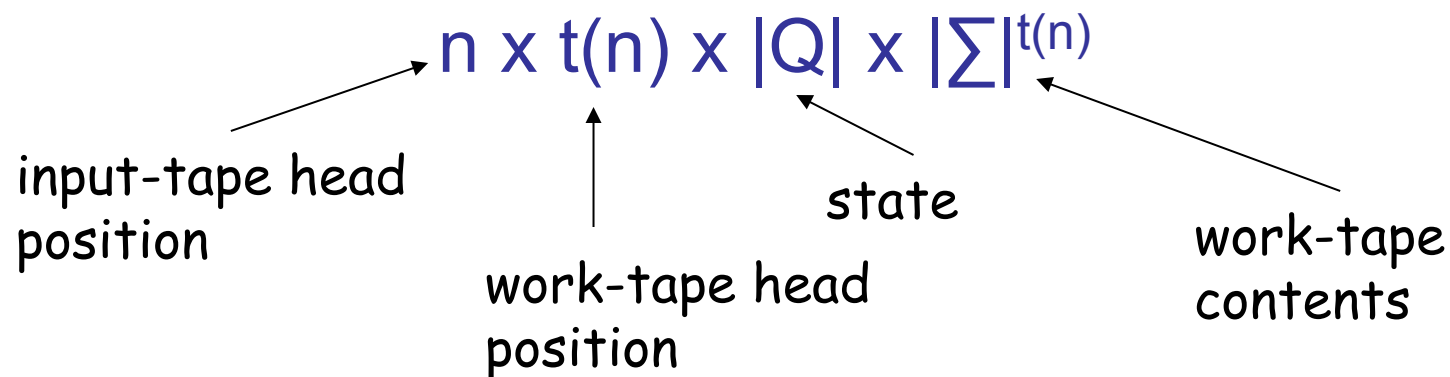
represented by string:

$$C = \sigma_1 \sigma_2 \dots \sigma_i q \sigma_{i+1} \sigma_{i+2} \dots \sigma_m$$

- start configuration for single-tape TM on input x : $q_{\text{start}} x_1 x_2 \dots x_n$

Relationships between classes

- easy to tell if C **yields** C' in 1 step
- **configuration graph**: nodes are configurations, edge (C, C') iff C **yields** C' in one step
- # configurations for a 2-tape TM (work tape + read-only input) that runs in **space** $t(n)$



Relationships between classes

- if $t(n) = c \log n$, at most

$$n \times (c \log n) \times c_0 \times c_1^{c \log n} \leq n^k$$

configurations.

- can determine if reach q_{accept} or q_{reject} from start configuration by exploring config. graph of size n^k (e.g. by DFS)
- Conclude: **$L \subseteq P$**

Relationships between classes

- if $t(n) = n^c$, at most

$$n \times n^c \times c_0 \times c_1^{n^c} \leq 2^{nk}$$

configurations.

- can determine if reach q_{accept} or q_{reject} from start configuration by exploring config. graph of size 2^{nk} (e.g. by DFS)
- Conclude: **PSPACE** \subseteq **EXP**

Relationships between classes

- So far:

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

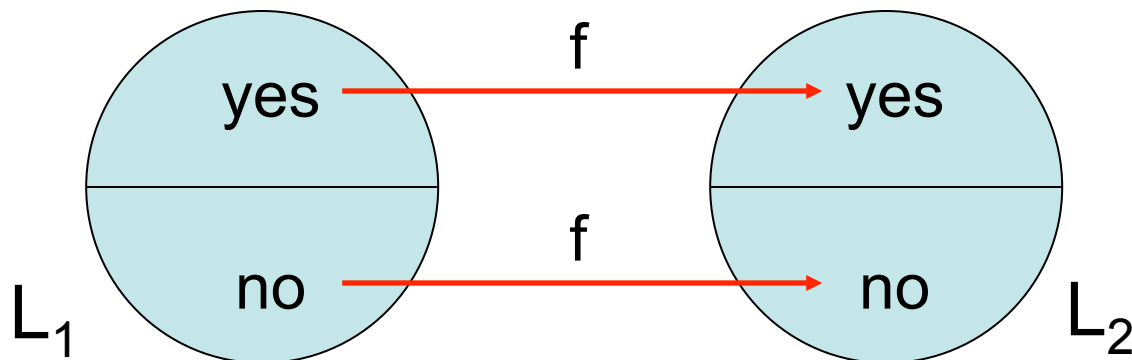
- believe all containments strict
- know $\mathbf{L \subsetneq PSPACE}$, $\mathbf{P \subsetneq EXP}$
- even before any mention of NP, two **major** unsolved problems:

$$\mathbf{L \stackrel{?}{=} P}$$

$$\mathbf{P \stackrel{?}{=} PSPACE}$$

A **P**-complete problem

- We don't know how to prove $L \neq P$
- But, can identify problems in **P** **least likely** to be in **L** using **P**-completeness.
- need stronger notion of reduction (why?)



A **P**-complete problem

- **logspace reduction**: f computable by TM that uses $O(\log n)$ space
 - denoted “ $L_1 \leq_L L_2$ ”
- If L_2 is **P**-complete, then L_2 in **L** implies **L** = **P** (homework problem)

A **P**-complete problem

- **Circuit Value (CVAL)**: given a variable-free Boolean circuit (gates \wedge , \vee , \neg , 0, 1), does it output 1?

Theorem: CVAL is **P**-complete.

- **Proof**:
 - already argued in **P**
 - L arbitrary language in **P**, TM M decides L in n^c steps

A P-complete problem

- **Tableau** (configurations written in an array) for machine M on input w :

w_1/q_s	w_2	...	w_n	...	—
w_1	w_2/q_1	...	w_n	...	—
w_1/q_1	a	...	w_n	...	—
⋮					⋮
—/q _a	—	...	—	...	—

- height = time taken = $|w|^c$

- width = space used $\leq |w|^c$

A P-complete problem

- Important observation: contents of cell in tableau determined by 3 others above it:

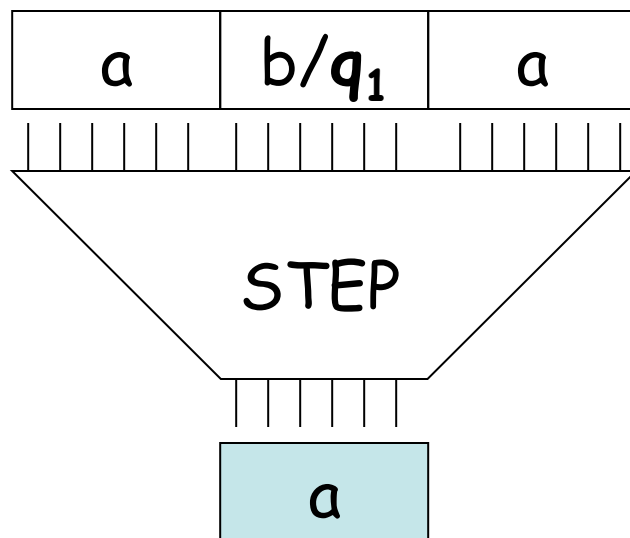
a/q_1	b	a
	b/q_1	

a	b/q_1	a
	a	

a	b	a
	b	

A P-complete problem

- Can build Boolean circuit STEP
 - input (binary encoding of) 3 cells
 - output (binary encoding of) 1 cell



- each output bit is some function of inputs
- can build circuit for each
- **size is independent of size of tableau**

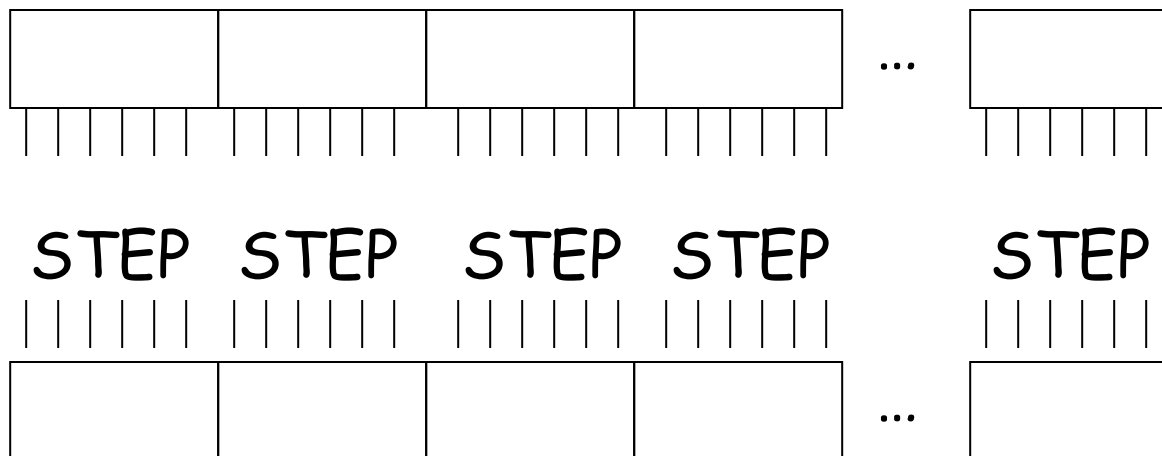
A P-complete problem

Tableau for
M on input

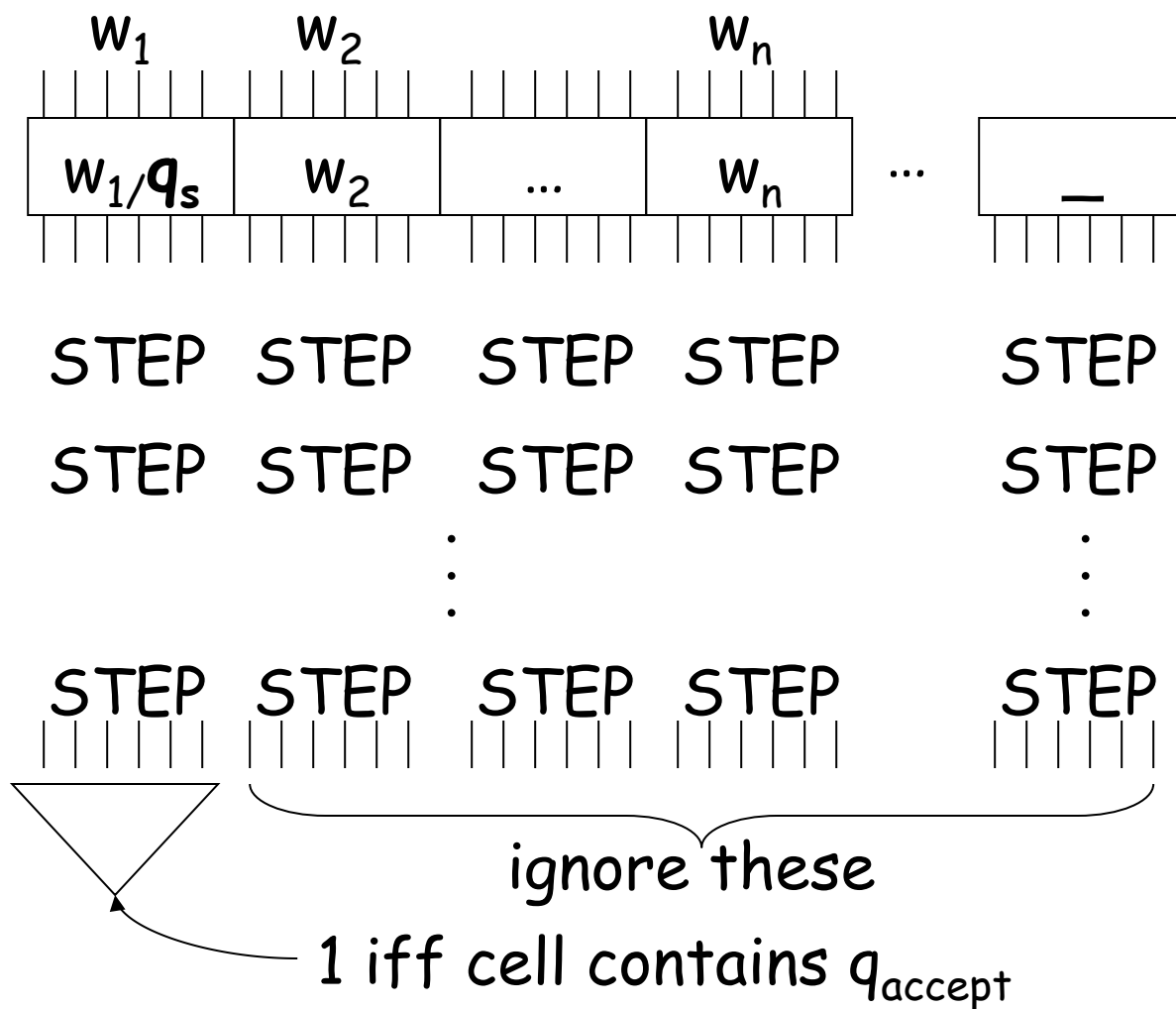
w

w_1/q_s	w_2	...	w_n	...	—
w_1	w_2/q_1	...	w_n	...	—
	⋮				⋮

- $|w|^c$ copies of STEP compute row i from $i-1$



A P-complete problem



This circuit $C_{M,w}$ has inputs $w_1 w_2 \dots w_n$ and $C(w) = 1$ iff M accepts input w .

logspace reduction

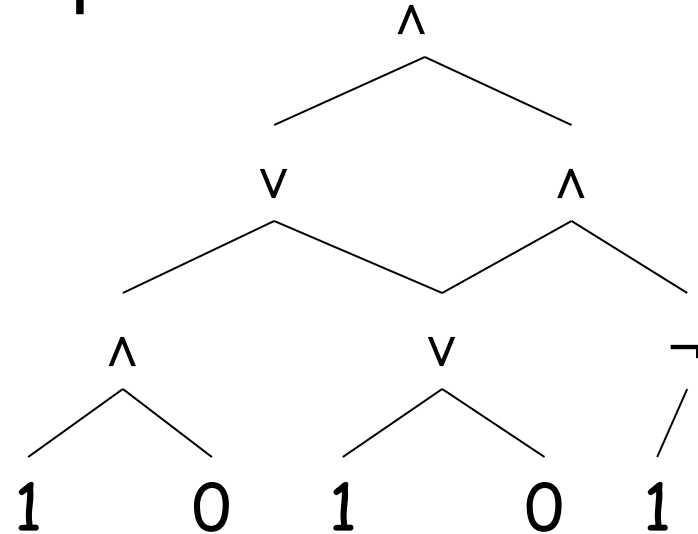
Size = $O(|w|^{2c})$

Answer to question

- Can we evaluate an n node Boolean **circuit** using $O(\log n)$ space?

- **NO!** (probably)

- **CVAL** in **L** if and only if **L = P**



Padding and succinctness

Two consequences of measuring running time as function of input length:

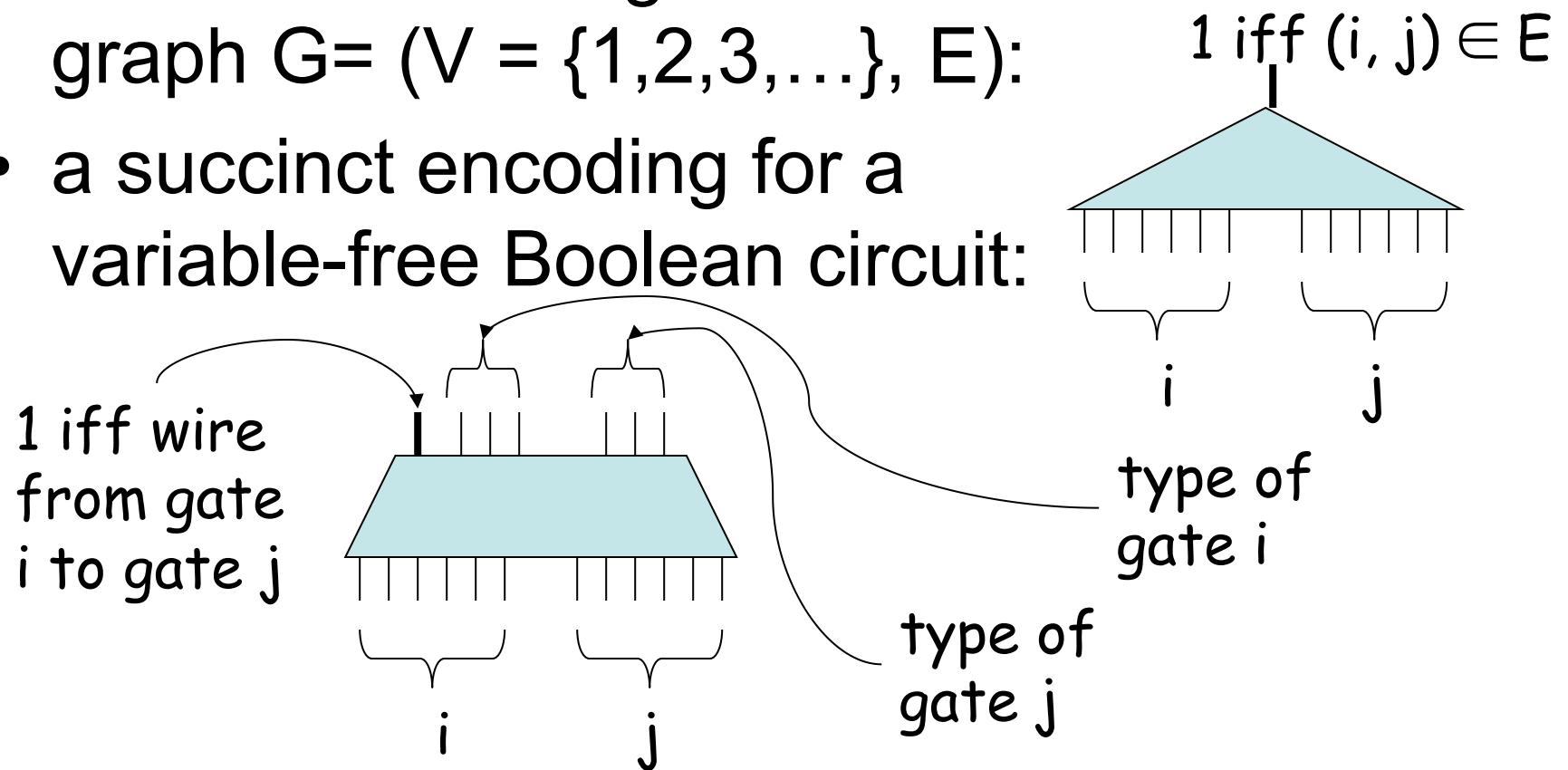
- “padding”
 - suppose $L \in \mathbf{EXP}$, and define
$$\text{PAD}_L = \{ x\#^N : x \in L, N = 2^{|x|^k} \}$$
 - TM that decides PAD_L : ensure suffix of N #s, ignore #s, then simulate TM that decides L
 - running time now polynomial !

Padding and succinctness

- converse (intuition only): “succinctness”
 - suppose L is **P-complete**
 - intuitively, some inputs are “hard” -- require full power of **P**
 - **SUCCINCT_L** has inputs encoded in different form than L , some exponentially shorter
 - if “hard” inputs are exponentially shorter, then candidate to be **EXP-complete**

Succinct encodings

- succinct encoding for a directed graph $G = (V = \{1, 2, 3, \dots\}, E)$:
- a succinct encoding for a variable-free Boolean circuit:



An **EXP**-complete problem

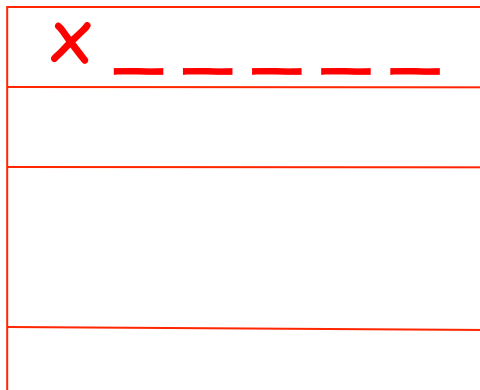
- **Succinct Circuit Value**: given a **succinctly encoded** variable-free Boolean circuit (gates \wedge , \vee , \neg , 0, 1), does it output 1?

Theorem: Succinct Circuit Value is **EXP**-complete.

- Proof:
 - in **EXP** (why?)
 - L arbitrary language in **EXP**, TM M decides L in 2^{n^k} steps

An **EXP**-complete problem

- **tableau** for input $x = x_1x_2x_3\dots x_n$:



height,
width 2^{n^k}

- Circuit C from CVAL reduction has size $O(2^{2^{n^k}})$.
- TM M accepts input x iff circuit outputs 1

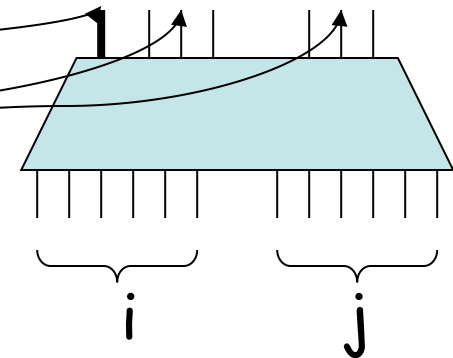
An **EXP**-complete problem

– Can encode C succinctly:

1 iff wire
from gate
i to gate j

type of
gate i

type of
gate j



- if i, j within single STEP circuit, easy to compute output
- if i, j between two STEP circuits, easy to compute output
- if one of i, j refers to input gates, consult x to compute output

Summary

- Remaining TM details: big-oh necessary.
- First complexity classes:

L, P, PSPACE, EXP

- First separations (via simulation and diagonalization):

$P \neq EXP, L \neq PSPACE$

- First major open questions:

$L \stackrel{?}{=} P$ $P \stackrel{?}{=} PSPACE$

- First complete problems:
 - CVAL is **P**-complete
 - Succinct CVAL is **EXP**-complete

Summary

