Complexity Classes

Note Title 3/31/2013

Consider Classes TIME($f(n)$) SPACE($f(n)$)

- How are these classes related to each other?

- How do we define robust classes?

  TIME (n) not robust in that it changes w/ small perturbations to the model of computation (e.g. k vs. $k+1$ tapes).

- What problems are in these classes (complete for the class).

Linear Speed-up Theorem:

Suppose TM M decides L in time $f(n)$
Then for any $\epsilon > 0$ $\exists$ TM M' that decides L in time $\epsilon f(n) + n + 2$

Leading constants are meaningless for time complexity classes

$$\text{TIME}(f(n)) = \text{TIME}(\epsilon f(n) + (1+\epsilon)n).$$

Improvements in hardware make leading constants meaningless.

$$L \in \text{TIME}(3n^2) \Rightarrow L \in \text{TIME}(n^2).$$

Big-oh notation is necessary given our model of computation.

$$f(n) = O(g(n)) \text{ if } \exists\, c, n_0 \text{ s.t. } \forall n \geq n_0$$
$$f(n) \leq c \cdot g(n).$$

A TM can not differentiate between TIME $(3n^2)$ + TIME (n^2)
We are interested in coarser grain distinctions, not the peculiarities of the Turing Machine.

Linear Speedup Theorem proof idea:

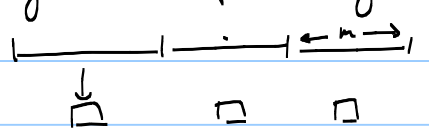
No set bound on the # of characters in Σ. Must be finite, but it can depend on ϵ.

M' will have one more tape than M.

$\Sigma_{new} = \Sigma_{old} \cup (\Sigma_{old})^m$ and many more states.

① Compress input onto a fresh tape.

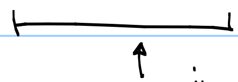
Every block of m symbols on input tape $\rightarrow$ 1 new symbol.



time: $n+1$

$\lceil \frac{n}{m} \rceil - 1$ to scan back

② Simulate M m steps at a time.



in m steps head can go to left or right of current block but not both.

- Move R L to read block to the right
- Remember contents in state
- R L again to update + reposition.

(or L R depending on contents of current block).

Can simulate m steps in ≤ 4 steps.

Running Time: $4\frac{f(n)}{m} + n + \lceil \frac{n}{m} \rceil$

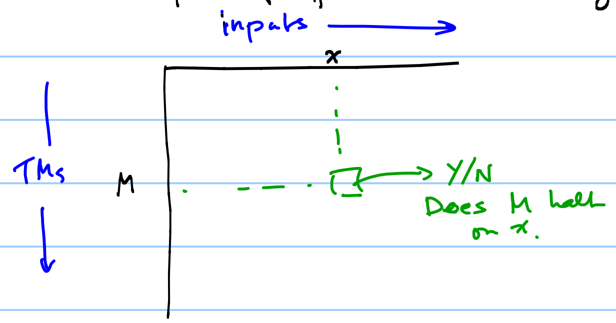
Choose $m = \lceil \frac{4}{\epsilon} \rceil$ $\hookrightarrow$ $\epsilon f(n) + n + \frac{\epsilon n}{4} + 1$.

How much additional time is needed in order to decide more languages?

When do we have $TIME(f(n)) \subsetneq TIME(g(n))$?

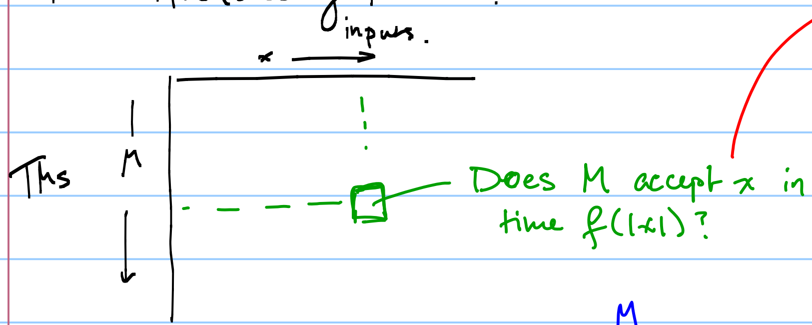
Need to construct a language <u>not</u> in TIME$(f(n))$
$\Rightarrow$ diagonalization.

Recall proof for the Halting Problem:



[the existence of H which always halts + tells you Y/N for each box allows you to construct H' which can not be in the table]

Time Hierarchy Theorem.



if the answer is no, it can be for one of two reasons:

1. M halts and rejects in time $f(|x|)$
2. M is still going after $f(|x|)$ steps.

If a TM M halts in $f(|x|)$ steps for every input, x, then the row corresponding to M will describe the language it decides (no premature halting).

Need TM SIM which takes input (M, x) and simulates M on x for $f(|x|)$ steps. SIM can determine the contents of the table.
There will be some overhead in the simulation, so SIM will compute in time $g(n)$.

Will use SIM to construct D which is not in the table. $\Rightarrow$ $L(D) \notin$ TIME$(f(n))$.

→ Output of D is the opposite of the diagonal entries of the table.

input length n

D on input $\langle M \rangle$, runs SIM on $\langle M, M \rangle$
outputs the opposite of SIM on input $\underbrace{(M,M)}_{\text{input length } = 2n}$.

running time of D: $g(2n)$.

Suppose $\bar{M}$ in time $f(n)$ decides $L(D)$ (language decided by D).

$$\bar{M}(\langle \bar{M} \rangle) = SIM(\langle \bar{M}, \bar{M} \rangle) \neq D(\langle \bar{M} \rangle).$$

but $\bar{M}(\langle \bar{M} \rangle) = D(\langle \bar{M} \rangle)$.

How fast can the simulation be done?
Given $f(n)$, what is $g(n)$? → $g(n) \leq [f(n)]^3$.

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

What is a "proper" function?

f is a proper complexity function if
$f(n) \geq f(n-1)$ $\forall n$
$\exists$ TM M that outputs $f(n)$ symbols on input 1^n
and runs in time $O(f(n)+n)$ + space $O(f(n))$

includes all functions that we will work with:
$\log n$ $\quad \sqrt{n}$ $\quad n^2$ $\quad 2^n$ $\quad n!$
if f + g are proper then so are:
$f+g$ $\quad fg$ $\quad f(g)$ $\quad f^g$ $\quad 2^g$ etc.

we will mostly ignore this issue

However: $\exists$ non-proper f:
$$TIME(f(n)) = TIME(2^{f(n)})$$

For proof of Time Hierarchy Theorem, we need only sketch how SIM works.

Space

SIM has 4 tapes Input $\langle M, x \rangle$ $|\langle M,x\rangle| = n$.

$k_m \ell_m f(|x|)$
- # tapes of M
- # bits to encode state/char of M.

1. Contents of M's tape + location of head.
2. M's state.
3. M's transition fcn.
4. $f(|x|)$ 1's (used for clock).

Initialize:
- Write $1^{f(|x|)}$ on 4^{th} tape.
- Copy $\langle M \rangle$ to 3^{rd} tape.
- Keep $\langle x \rangle$ on tape 1.
- Write q_{start} on tape 2.

$O(\ell_m k_m f(|x|))$ Copy current symbol of tape 1 onto tape 2.

$O(|\langle M \rangle|)$ 2nd tape now has encoding of $(q, \sigma_1 \cdots \sigma_k)$ $\sigma_i \in \Sigma$.

$O(|\langle M \rangle|)$ Match symbols to find correct rule to apply from tape 3.

Change state on tape 2.

Update current character on tape 1.

Update head position → could require $k_m \cdot$ length of tape 1 for moving over tape symbols

Advance clock by one.

One step of M: $O(\ell_M k_M^2 f(|x|))$ steps.

Run time of SIM $O(\ell_m k_M^2 f^2(|x|))$ ← $O(f^3(n))$

length of encoding $\langle M, x \rangle = n$

$|\Sigma|^k \, |Q| \cdot \ell_m$

//

Does more space allow us to decide more languages?

Theorem: (Space Hierarchy Theorem)
For every proper complexity function $f(n) \geq \log(n)$,
$$\mathrm{SPACE}(f(n)) \subsetneq \mathrm{SPACE}(f(n) \log f(n))$$

Proof is similar to the Time Hierarchy theorem.

Here are some of the most common complexity classes that we will study:

$$\mathcal{L} = \mathrm{SPACE}(\log n)$$
$$\mathrm{PSPACE} = \bigcup_{k \geq 1} \mathrm{SPACE}(n^k)$$
$$\mathrm{P} = \bigcup_{k \geq 1} \mathrm{TIME}(n^k)$$
$$\mathrm{EXP} = \bigcup_{k \geq 1} \mathrm{TIME}(2^{n^k})$$

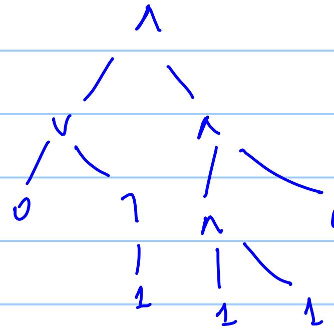
input on read only tape
log n refers to # cells used on a work tape.

Here are some problems in these classes:

$\underline{\underline{\mathcal{L}}}$: FVAL (function evaluation)

Input: boolean formula (each variable appears only once)
& a truth assignment for variables.
Does the formula evaluate to true?

boolean formula expressed as a tree:



PSPACE: 2-person games

e.x. Generalized Geography

Input: directed graph.

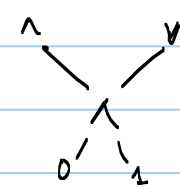
players take turns selecting an outgoing edge from the current node. Current node becomes the node that is reached from the selected edge. Player loses if there is no out-going edge to select.

Question: Does Player 1 have a winning strategy?

P: CVAL, linear programming, max flow.

↳ given a boolean circuit & an assignment to the inputs what are the values of the outputs?

Difference between a circuit and a formula is that circuits can have fan-out > 1.



EXP: Boolean SAT, NP, more....

What are some of the relationships between these classes?

From the Time Hierarchy Theorem:

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

$$\Rightarrow \quad P \subsetneq EXP.$$

From the space hierarchy theorem:

$$\mathcal{L} = \text{SPACE}(\log n) \subsetneq \text{SPACE}(\log^2 n) \subseteq \text{PSPACE}$$

$$\mathcal{L} \subsetneq \text{PSPACE}.$$

$P \subseteq \text{PSPACE}$.

What about $\mathcal{L}$ vs P?

PSPACE vs. EXP?

We shall see:

$\mathcal{L} \subseteq P$

$\text{PSPACE} \subseteq \text{EXP}$

but we don't know whether they are equal. (conjectured not equal).

The entire state or "configuration" of a Turing Machine can be described by: contents of tape
location of head.
state.

Can be described by string: $\sigma_1 \sigma_2 \sigma_3 \cdots \sigma_{i-1}\, q\, \sigma_i \cdots \sigma_m$

location of head.

↳ for multi-tape TMs, tape contents are separated by #.

C_{start} : $q_{start}\, x_1 x_2 \cdots x_n$

Easy to determine if $C \rightarrow C'$ in one step.

Configuration graph : nodes $\leftrightarrow$ set of all configurations.
Edge (C, C') if C yields C' in one step.
(out degree ≤ 1).

The # of configurations of a 2-tape TM with one read-only input tape and one work tape that uses $\leq t(n)$ cells:

$$\underbrace{n \times t(n)}_{\text{head positions}} \times |Q| \times |\Sigma|^{t(n)}$$

for $t(n) = \log n$
configs $\leq n^k$

Alg: Calculate the # of configurations (or any poly upper bound).
Start w/ C_{start} + follow the path of configurations.
Stop if # steps $\geq$ # configurations, or if a halting configuration is reached

→ there must have been a repetition which means an infinite loop has been reached.

$\Rightarrow \mathcal{L} \subseteq P$.

If $t(n) = n^c$ # configs $\leq n \times n^c \times c_0 \times c_1^{n^c} \leq 2^{n^k}$ for some k.

c_0: # states

c_1: # chars in Σ.

configurations is exponential in n.
Same idea as before but now the timer is 2^{n^k}

So we have: $\mathcal{L} \subseteq P \subseteq PSPACE \subseteq EXP$

Know: $\mathcal{L} \subsetneq PSPACE$

but we don't know if $\mathcal{L} \subsetneq P$ or $P \subsetneq PSPACE$ or both.

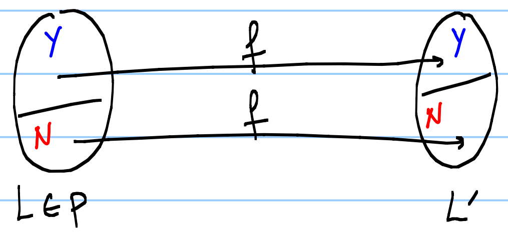
(at least one of the inequalities must hold).

General belief is that all containments are strict.

We don't know if $\mathcal{L} \subsetneq P$ but we can identify a problem in P that is least likely to be in $\mathcal{L}$.

We want a P-complete problem L'

$\forall L \in P \quad L \propto L'$ so that if $L' \in \mathcal{L}$ then $\forall L \in \mathcal{L}$



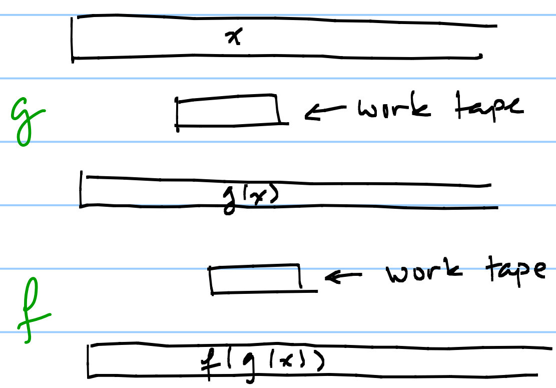
To compute L on input x:

Compute $f(x)$ — needs to be in logspace too.

then run logspace alg for L' on $f(x)$.

We need logspace reductions!

Not obvious how to compose logspace reductions.



We want a logspace computation that goes from x directly to $f(g(x))$ but we cannot write down $g(x)$ in its entirety.

Solution: Simulate M_f. Whenever it needs to read the i^{th} bit of the input:

need to keep a register with location of read head in binary.

- Remember current state $O(\log|Q_{M_f}|)$ bits.
- Restart M_g until the i^{th} bit of the output is written. (Don't write the other output bits). Note since output tape is write-only, we can assume it's head never moves left.
- Simulate next step of M_f.

Space used: Work tape of M_g.
Work tape of M_f.
State of M_f
Counter for i $\Rightarrow O(\log n)$.

$L_1 \in P$ $\quad L_1 \propto L_2$ by logspace reduction.

if $L_2 \in \mathcal{L} \Rightarrow L_1 \in \mathcal{L}$.

A P-complete problem

Given a variable-free boolean circuit, does it evaluate to 1?

Dfn of a circuit:

Directed-Acyclic-Graph (DAG)

with exactly one sink.

Every internal node is labelled with a gate. $\wedge, \vee$ have in-degree 2, $\neg$ has in-degree one.

Every Source (no in-degree) is labelled with a 0, 1 or Varible. (in our case, no variables).

CVAL $\in$ P (straightforward)

Will show $\forall L \in P$ $\quad L \propto_{\mathcal{L}}$ CVAL

logspace reduction.

Since $L \in P$ $\exists$ TM M that decides L in n^k steps.

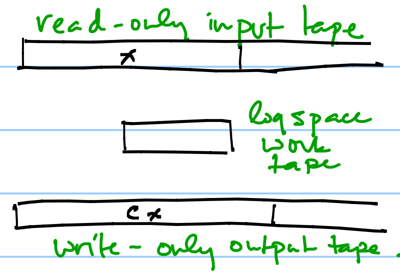
→ we can assume w.l.o.g. that when M halts, the head is in left-most cell. (will be convenient later).

We will describe a logspace TM R → depends on M only (not x).

$\wedge \ \neg \ \vee \ 0 \ 1$

(gate label, gate type)
(gate label, gate label)

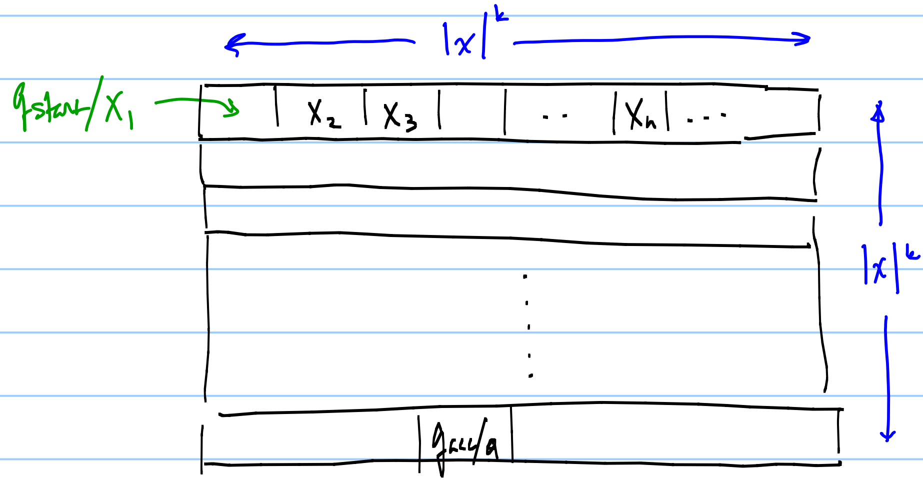
Input to R: x



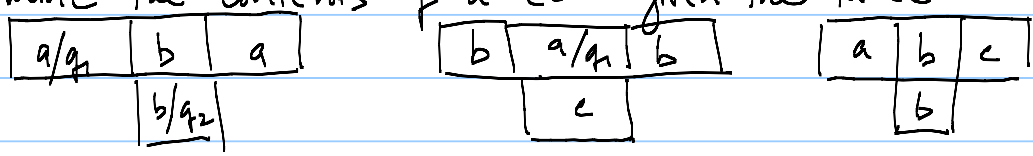
Output of R: C_x – variable-free circuit

$x \in L \iff C_x$ evaluates to 1.

Examine the computation that M performs on input x.
We can construct a tableau showing the history of the computation for input x.
Each row corresponds to a configuration at a particular time step of the algorithm. Contents of a cell is $\sigma \in \Sigma$ or $(q, \sigma) \in Q \times \Sigma$



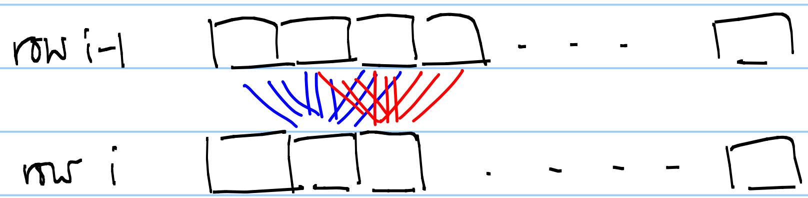
We can determine the contents of a cell given the three above it:



$\delta(q_1, a) \to (c, q_2, R)$

Can build a constant-sized boolean formula STEP
input: binary encoding of 3 cells.
output: binary encoding of cell below.
Depends only on M, not input x. hard-coded into R's rules.

Can build a poly-sized circuit using STEP as a component that simulates M's computation on x.
to get row i from row $i-1$, have $|x|^c$ copies of STEP.



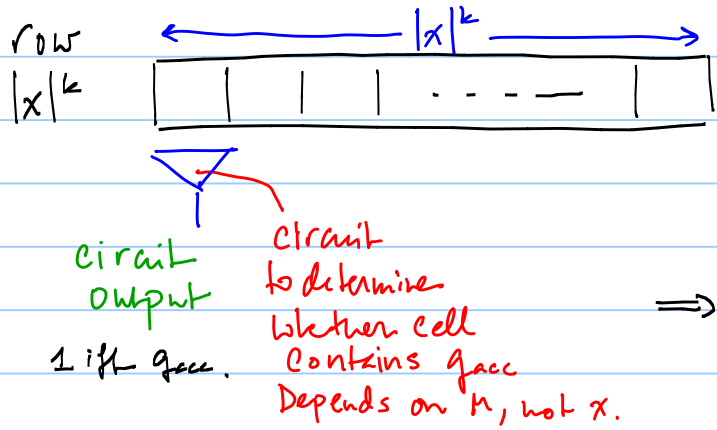
There are $|x|^{2k}$ copies of STEP.

To output these gates and connections, need a counter of size $O(\log(|x|^{2k}))$ to keep track of which copy is being output. Gate label $\langle$row, col, gate in STEP$\rangle$.

Circuit input: row 1 $\boxed{q_{start}/x_1}\boxed{x_2}\boxed{x_3}\cdots\boxed{x_n}\boxed{\sqcup}\cdots\boxed{\sqcup}$

Encoding of each x_i and $\sqcup$ in binary are the inputs to first row.
$(\langle 1, i, j\rangle, x_{ij})$ — j^{th} bit in encoding of x_i: $1 \le j \le \log|\Sigma| + \log|Q|$.

Circuit Output:



Finally we need gates to determine whether M accepts

$\Longrightarrow$ M accepts $x \iff x \in L$
$\Updownarrow$
ckt evaluates to 1.

Can we evaluate a ckt in $O(\log n)$ space?
Probably NO.

CVAL $\in \mathcal{L}$ iff $\mathcal{L} = P$.

Padding + Succinctness

Consequence of measuring complexity of input length:
- artificially expand input $\Rightarrow$ running time decreases.
- shrink input $\Rightarrow$ running time becomes larger.

Padding: Suppose $L \in EXP$ ($L \in TIME(2^{n^k})$ for some k).

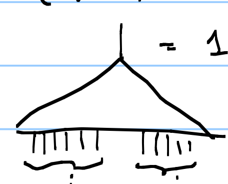
$$PAD_L = \{ x \#^N \mid x \in L \quad N = 2^{|x|^k} \}$$

TM that decides PAD_L
→ Verify that the padding (string of #'s) has the correct length.
→ Run TM for L on x (ignore #'s).

Running time is linear in input length!

Conversely: Intuition: Suppose L is P-complete.
$SUCCINCT_L$ encodes inputs of L exponentially shorter.
If hard instances of L are exp. shorter, then it is a candidate to be EXP-complete.

Succinct encodings:
Instead of specifying a graph by enumerating edges and vertices,
Specify n = # vertices (written in binary)
Give a circuit



= 1 iff $(i,j) \in E$

circuit size poly in
inputs = $2 \log n$.

Not all graphs can be specified succinctly but may be
Some problems are still hard even when input graphs are restricted to graphs that can be specified this way.

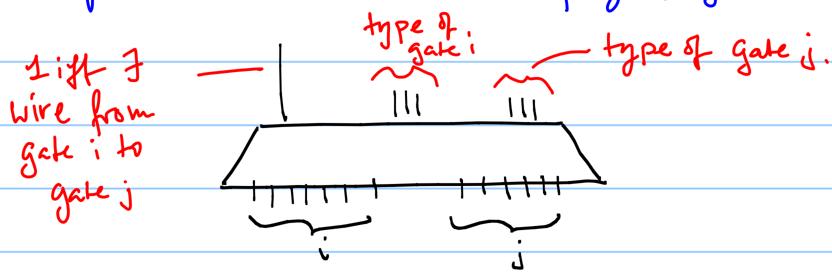
Succinct encoding of instances of CVAL (variable-free ckt).

Small circuit that encodes a big circuit.

① m = # nodes in big circuit (written in binary).

② Specification of a ckt that takes $2\log m$ inputs.

size of small ckt is $O(\text{poly}(\log m))$



Succinct CVAL : given a succinctly encoded variable-free Boolean ckt, does it output 1?

Theorem: SUCCINCT CVAL is EXP-complete.

SUCCINCT-CVAL $\in$ EXP. m = # nodes in big circuit.

Run through all k : $1 \le k \le m$

output type of gate k.

Run through all (k,ℓ) $1 \le k,\ell \le m$

output (k,ℓ) if there is an edge from (k,ℓ).

Now we have an explicit description of the circuit (size m^2).

Solve CVAL in time poly(m).

Note, input length $\ge \log m$.

$n \ge \log m$ $\quad$ $\text{poly}(m) \le m^k \le 2^{n \cdot k}$

Now show $\forall$ $L \in EXP$, $L \propto$ SUCCINCT CVAL

$\leadsto$ M decides L in 2^{n^k} steps.

$$x \xrightarrow{\text{computable in time poly}(|x|).} \text{Succinct description of } C_x$$

Size of C_x $2^{|x|^k}$

Encoding of C_x $|x|^k$ bits.

$x \in L \iff C_x$ evaluates to true.

C_x is the same "tableau" ckt as in proof that CVAL is P-complete:

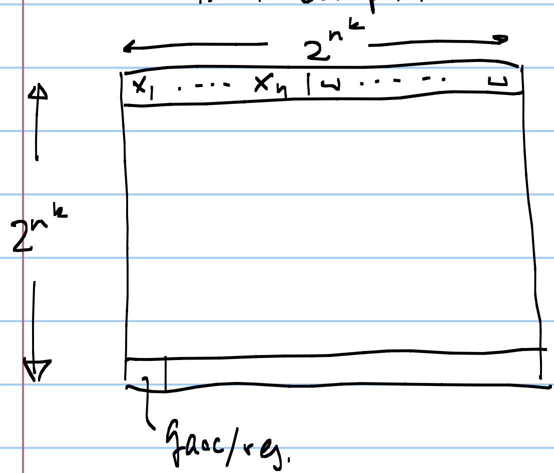


Tableau represents computation of M on input x.

Can express tableau as a ckt.

Input: binary encoding of $x \sqcup 2^{n^k - n}$

One copy of STEP for each cell in rows $2 \ldots 2^{n^k}$

TM M accepts on input x iff C_x evaluates to 1.

How to express C_x using only $O(|x|^c)$ bits?

\# gates in step $= S$.

\# bits to encode a cell $= r$ $\log(|\Sigma| + |\Sigma| \cdot |Q|)$

Gate label: (row, col, i)

Give a decision procedure to determine type of a gate

(L circuit)

poly in length of encoding of (row, col, i)

poly ($\log r$)

for row = 0, gates are input gates $1 \le i \le r$
if col $\le n$ (row, col, i) is i^{th} bit of encoding of x_i
if col $> n$ (row col, i) is i^{th} bit of encoding of ⊔
for row ≥ 1 gates are gates in STEP $1 \le i \le s$.
type of (row, col, i) depends only on $i \Rightarrow O(s)$.

How to determine edges.
Outputs of (row−1, col−1, ·) (row−1, col, ·) (row−1, col+1, ·)
go to inputs of (row, col, ·).
determining of row' = row−1
col' = col ±1, 0
is a poly-time procedure

Finally can hard-code outputs of $(2^{|x|^k}, 0, -)$

=

SUMMARY: First complexity classes:

$\mathcal{L}$, P, PSPACE, EXP $\mathcal{L} \subseteq P \subseteq PSPACE \subseteq EXP$.

1^{st} separation via diagonalization (hierarchy theorem)
$P \neq EXP$ $L \neq PSPACE$

1^{st} major open questions:
$\mathcal{L} \stackrel{?}{=} P$ $P \stackrel{?}{=} PSPACE$

Complete Problems: CVAL is P-complete
Succ-CVAL is EXP-complete.