10/27/2015

Concrete Complexity Theory

- with TMs, we don't have any "concrete" lower bounds (like, SAT needs $\Omega(n^2)$ time).
- looking at circuits (or restrictions of circuits), we'll try to get more precise lower bounds.

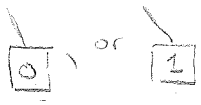
Recall: P/poly is non-uniform analog of P

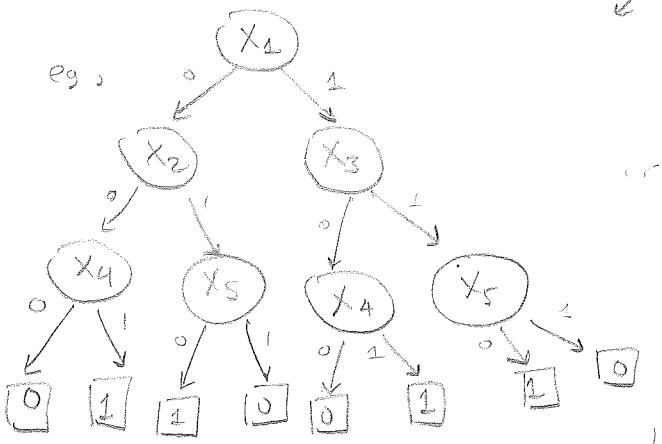
Today: "L/poly", branching programs, which are the nonuniform analog of L.

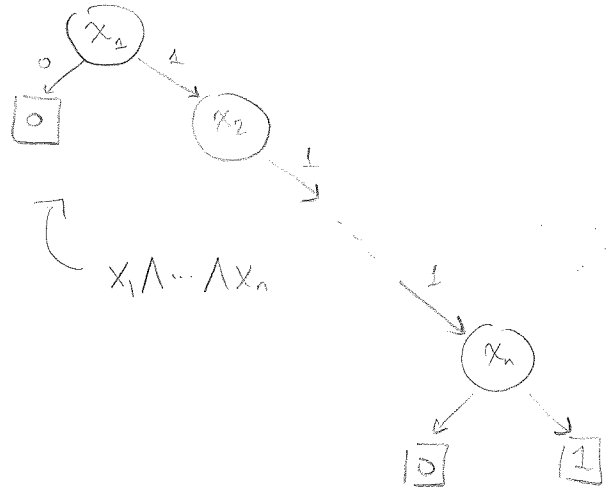
Along the way:
- formulae
- bdd depth circuits
- decision trees

Decision Trees

Given a fn $f: \{0,1\}^n \to \{0,1\}$

Definition by example: A decision tree is a rooted binary tree, with internal nodes  or leaves 





Clearly, every fn has a depth n decision tree, and every fn has a decision tree of size 2^n.

DT-depth(f) is the depth of the shallowest decision tree for f

DT-size(f) is the size of the smallest decision tree for f

10/27/2015

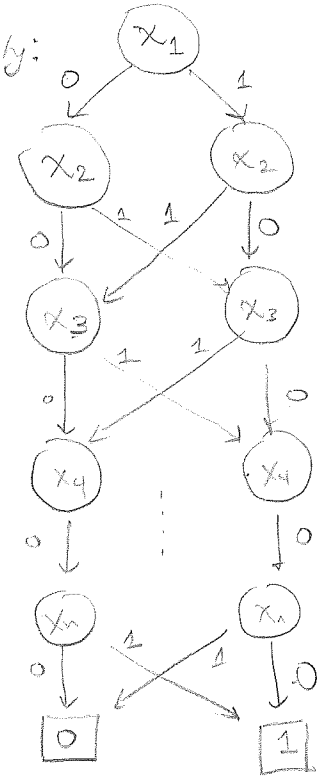
Decision trees are a bit wasteful — as in the previous example, we might have the same subtree copied twice.

Branching programs are decision trees where we can "re-use" nodes.

Formally, a branching program on n-variables is
- a directed acyclic graph, with
- one source
- two sinks, $\boxed{0}$ and $\boxed{1}$
- each non-sink is labelled by an "x_i", and has exactly two outgoing edges.

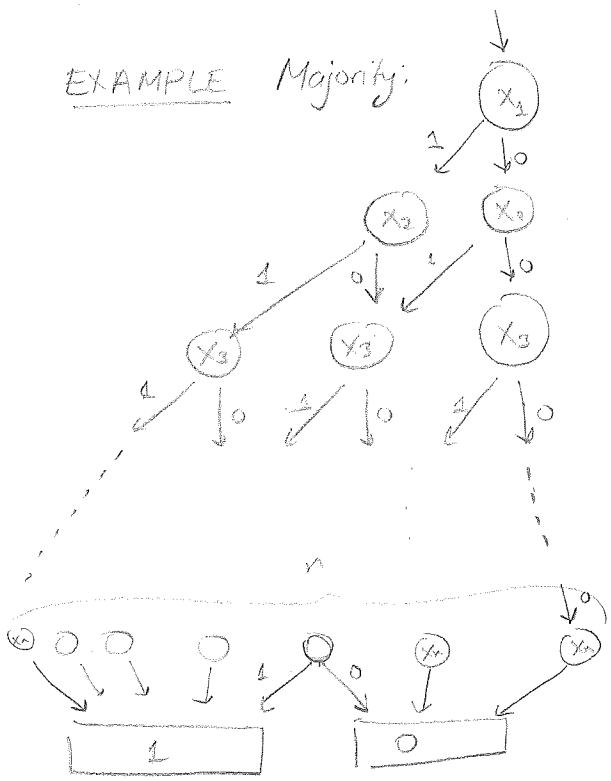
A branching program computes a fn by starting at the source and following until you get to a sink.

EXAMPLE: Parity:



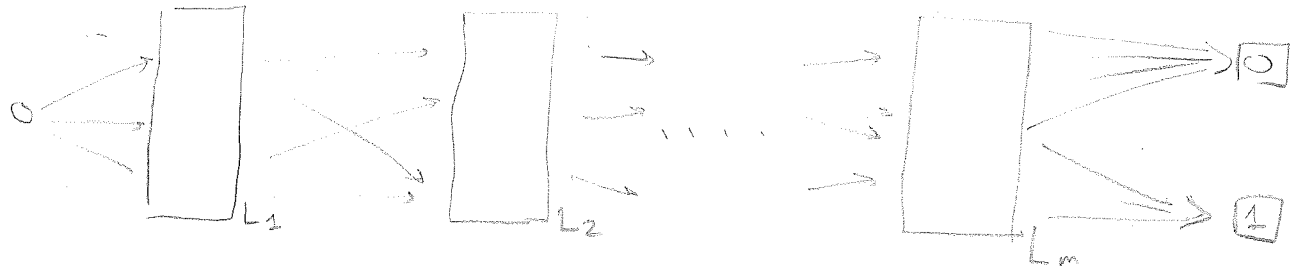
much smaller than the size 2^n decision tree!

EXAMPLE Majority:



10/27/2015

Both examples are "LAYERED BRANCHING PROGRAMS"



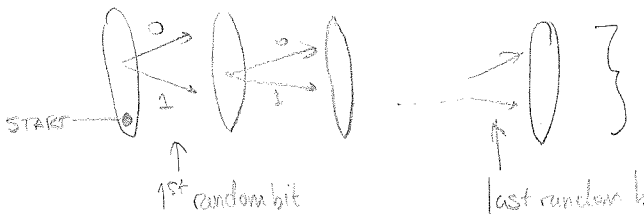
Nodes are arranged in LAYERS, and all edges go from L_i to L_{i+1}.

Two important things:
- how many layers (this is like time)
- width of the layers (this is like memory)

} The SIZE of the branching program is the total # of nodes.

eg, - Majority had width n
- Parity had width 2

Recall — we've seen this model before to "fool" RL machines w/ Nisan's generator:



} #states was $\leq 2^{O(S)}$, where S was the space of the machine.

Log space + Branching Programs.

Thm If $A \in$ LOGSPACE, then A is decided by a family of poly(n)-sized branching programs.

aka, you have $B_1, \ldots, B_n, \ldots$ s.t. $\forall x \in \{0,1\}^*$, $B_n(x) \Leftrightarrow x \in A$.

Why? Idea:
(1) Nodes $\longleftrightarrow$ configurations (tape, state, head locations)
(2) Labels $\longleftrightarrow$ value under read head
(3) Edges $\longleftrightarrow$ transition fn for the TM

at the end, all accepting configs go to $\rightarrow$ [1], all rejecting go to $\rightarrow$ [0]. ∎

10/27/2015

These feel like easier objects to think about than TMs.

So to prove NP ≠ L, we "just" have to show SAT doesn't have $n^{O(1)}$-sized BP$_5$.

(Best lower bd for SAT is currently $\Omega(n^3/\log^2(n))$ ...) ← from 1966

Proving lower bounds on branching programs seems hard — maybe it will get easier if we consider only SMALL-WIDTH, LAYERED branching programs.

(say, constant)
(say, 5.)

EXERCISE Every fn can be computed by a width-3 branching program. ← might be 4...

Return to Majority fn —

Can we do better than width n?
Can we prove a lower bound?

~~CLAIM~~ Majority requires super-poly-sized BP to compute in $O(1)$-width.

Surprisingly, this turns out to be FALSE!

Thm (Barrington, 1980's) Majority admits width-5 polysized BP$_5$.

↳ In fact, this is true for all fns w/ poly-sized FORMULAE.

Formulae Def. by example:

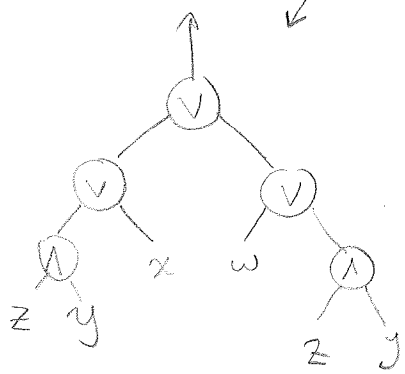
$$(x \vee (y \wedge z)) \vee (w \vee (y \wedge z))$$

↑ in a circuit, you could re-use this bit, but w/ formulae you can't.

Def. by def:

A formula is a tree, directed toward the root, w/ fanout 1. The leaves have variable names or negations. The internal node has $\wedge$ and $\vee$, fanin 2.

SIZE = # leaves
DEPTH = depth of tree



So formulae are bigger than circuits, since we can't re-use things in formulae.

10/27/2015

MAJORITY does have polysized formulae: in HW 1, you showed MAJ has size $O(n)$, depth $O(\log(n))$ circuits.

Any depth $\log(n)$ circuit can be turned into a formula of poly(n) size, depth $\log(n)$.

$\Rightarrow$ MAJ has poly(n)-size, $\log(n)$-depth formulae.

So far, we've seen:

Circuits
Branching programs
Formulae

} how do these compare?

EXERCISE: Let $f: \{0,1\}^n \to \{0,1\}$

(1) $C(f) \leq O(BP(f))$

$C(f)$: size of smallest circuit for f w/ fanin 2, $\wedge, \vee, \neg$ gates

(2) $BP(f) \leq O(L(f))$

$L(f)$: formula size w/ $\wedge, \vee$ gates

Define

$$\text{FOR-POLY} = \{ A \subseteq \{0,1\}^* \mid A \text{ is decided by a family of poly-size formulae} \}$$

Fact: Any formula of size s can be rebalanced to have depth $O(\log(s))$, size $\text{poly}(s)$.

[the analog for circuits is not known and not believed to be true]

More definitions!

$$NC^k = \{ A \mid A \text{ is decided by a family of polysize circuits of depth } O(\log^k(n)) \}$$

"Nick's Class"

We will mostly talk about NC^1, NC^2.

Fact: (Barrington's thm, restated) $\text{FOR-POLY} = NC^1$

aka, $A \in NC^1 \iff A$ is decided by a family of polysized formulae

10/27/2015

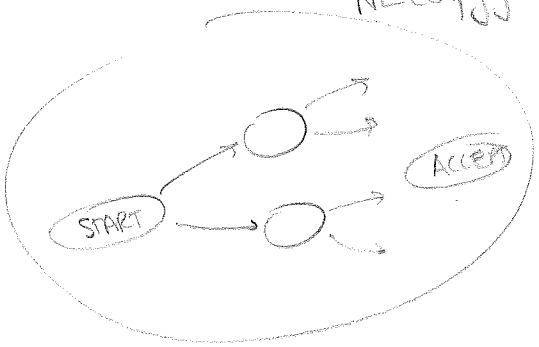
Pf. of Barrington's thm deferred until Thursday.

Some notes:

- Why is NC^2 important? Lots of linear algebra can be solved in NC^2.
- uniform-$NC^1 \subseteq L \subseteq NL \subseteq NC^2$
  (there's an efficient way to get the n^{th} circuit)
- It is believed that $P \not\subseteq$ uniform-NC^k $\forall k$ — that is, some things are NOT parallelizable.
- NP vs NC^1 is open.
  PH vs NC^1 "should be open"?

For the last 5 min, $NL \subseteq NC^2$

For a language in NL, we have NL configgraph G_x^M (for machine M, input x).



Want to know, is there a path START → ACCEPT.
We saw that this is NL-complete — now we just want to show we can do it with a shallow circuit.

Let A be adjacency matrix of G_x^M.

So we want to know, is $(A^n)_{START,\,ACCEPT} > 0$?

Or, is $\underbrace{(A \odot A \odot \cdots \odot A)}_{\text{Boolean matrix mult: } \sum \mapsto \vee,\ \text{and } \prod \mapsto \wedge}_{START,\,ACCEPT} = 1$?

You can do $A \odot A$ in log depth.

Recursively break this down to get a $\log^2$-depth circuit for the whole thing.

Caveat: the ENTRIES of A depend on the input x, and you need to verify that these are nice.