Two startling theorems

• Strongly believe $P \neq NP$
• Nondeterminism seems to add enormous power
• For space: Savitch '70:
  \[ \text{NPSPACE} = \text{PSPACE} \]
  and
  \[ \text{NL} \subseteq \text{SPACE}(\log^2 n) \]

Two startling theorems

• Strongly believe $NP \neq \text{coNP}$
• Seems impossible to convert existential into universal
• For space: Immerman/Szelepscényi '87/88:
  \[ \text{NL} = \text{coNL} \]

Savitch’s Theorem

Theorem: $\text{STCONN} \subseteq \text{SPACE}(\log^2 n)$

• Corollary: $\text{NL} \subseteq \text{SPACE}(\log^2 n)$
• Corollary: $\text{NPSPACE} = \text{PSPACE}$

Proof of Theorem

– Input: $G = (V, E)$, two nodes $s$ and $t$
– Recursive algorithm:
  
  ```
  /* return true iff path from $x$ to $y$ of length at most $2^i$ */
  PATH(x, y, i)
  if $i = 0$ return ($x = y$ or $(x, y) \in E$)
  /* base case */
  for $z \in V$
    if PATH(x, z, i-1) and PATH(z, y, i-1) return(true);
  return(false);
  ```

Proof of Theorem

– Answer to STCONN: $\text{PATH}(s, t, \log n)$
– Space used:
  • $(\text{depth of recursion}) \times (\text{size of stack record})$
  • $\text{depth} = \log n$
– Claim stack record: “$x, y, i$” sufficient
  • $\text{size } O(\log n)$
– When return from PATH($a, b, i$) can figure out what to do next from record ($a, b, i$) and previous record
Nondeterministic space

- Robust nondeterministic space classes:
  \[ \text{NL} = \text{NSPACE}(\log n) \]
  \[ \text{NPSPACE} = \bigcup_k \text{NSPACE}(n^k) \]

Second startling theorem

- Strongly believe \( \text{NP} \neq \text{coNP} \)
- Seems impossible to convert existential into universal
  \[ \text{for space: Immerman/Szelepscényi '87/'88:} \]
  \[ \text{NL} = \text{coNL} \]

I-S Theorem

**Theorem:** \( \text{ST-NON-CONN} \in \text{NL} \)

- Observation: given count of # nodes reachable from s, can solve problem
  - for each \( v \in V \), guess if it is reachable
  - if yes, guess path from s to v
    - if guess doesn't lead to v, reject.
    - if \( v = t \), reject.
    - else counter++
  - if counter = count accept

- Every computation path has sequence of guesses...
  - Only way computation path can lead to accept:
    - Correctly guessed reachable/unreachable for each node \( v \)
    - Correctly guessed path from s to v for each reachable node v
    - Saw all reachable nodes
    - t not among reachable nodes
I-S Theorem

- \( R(i) = \) # nodes reachable from \( s \) in at most \( i \) steps
- \( R(0) = 1: \) node \( s \)
- we will compute \( R(i+1) \) from \( R(i) \)
- only \( O(\log n) \) space and nondeterminism

\[ R(i) = R(2) = 6 \]

I-S Theorem

- Outline: in \( n \) phases, compute \( R(1), R(2), R(3), \ldots R(n) \)
- only \( O(\log n) \) bits of storage between phases
- in end, lots of computation paths that lead to reject
- only computation paths that survive have computed correct value of \( R(n) \)
- apply observation.

I-S Theorem

- computing \( R(i+1) \) from \( R(i) \):
  - Initialize \( R(i+1) = 0 \)
  - For each \( v \in V \), guess if \( v \) reachable from \( s \) in at most \( i+1 \) steps

I-S Theorem

- if "yes", guess path from \( s \) to \( v \) of at most \( i+1 \) steps. Increment \( R(i+1) \)
- if "no", visit \( R(i) \) nodes reachable in at most \( i \) steps, check that none is \( v \) or adjacent to \( v \)
  - for \( u \in V \) guess if reachable in \( \leq i \) steps; guess path to \( u \); counter++
  - KEY: if counter \( \neq R(i) \), reject
  - at this point: can be sure \( v \) not reachable

I-S Theorem

- two types of errors we can make
  - (1) might guess \( v \) is reachable in at most \( i+1 \) steps when it is not
    - won’t be able to guess path from \( s \) to \( v \) of correct length, so we will reject.
    - "easy" type of error

I-S Theorem

- (2) might guess \( v \) is not reachable in at most \( i+1 \) steps when it is
  - then must not see \( v \) or neighbor of \( v \) while visiting nodes reachable in \( i \) steps.
  - but forced to visit \( R(i) \) distinct nodes
  - therefore must try to visit node \( v \) that is not reachable in \( \leq i \) steps
  - won’t be able to guess path from \( s \) to \( v \) of correct length, so we will reject.
  - "easy" type of error
Summary

• nondeterministic space classes \(\mathbf{NL}\) and \(\mathbf{NPSPACE}\)

• ST-CONN \(\mathbf{NL}\)-complete

Summary

• Savitch: \(\mathbf{NPSPACE} = \mathbf{PSPACE}\)
  – Proof: ST-CONN \(\in \mathbf{SPACE}(\log^2 n)\)
  – open question: \(\mathbf{NL} = \mathbf{L}\)?

• Immerman/Szelepcsényi : \(\mathbf{NL} = \mathbf{coNL}\)
  – Proof: ST-NON-CONN \(\in \mathbf{NL}\)

Introduction

Power from an unexpected source?
• we know \(\mathbf{P} \neq \mathbf{EXP}\), which implies no poly-time algorithm for Succinct CVAL
• poly-size Boolean circuits for Succinct CVAL ??

Does \(\mathbf{NP}\) have linear-size, log-depth Boolean circuits ??

Outline

• Boolean circuits and formulas
• uniformity and advice
• the \(\mathbf{NC}\) hierarchy and parallel computation
• the quest for circuit lower bounds
• a lower bound for formulas

Boolean circuits

• circuit \(C\)
  – directed acyclic graph
  – nodes: AND (\&); OR (\lor);
  – NOT (\neg); variables \(x_1, x_2, \ldots, x_n\)

• \(C\) computes function \(f:\{0,1\}^n \rightarrow \{0,1\}\) in natural way
  – identify \(C\) with function \(f\) it computes
Boolean circuits

- **size** = # gates
- **depth** = longest path from input to output
- **formula (or expression)**: graph is a tree

- every function \( f : \{0,1\}^n \rightarrow \{0,1\} \) computable by a circuit of size at most \( O(n^2) \)
  - AND of \( n \) literals for each \( x \) such that \( f(x) = 1 \)
  - OR of up to \( 2^n \) such terms

Circuit families

- circuit works for specific input length
- we’re used to \( f : \Sigma^* \rightarrow \{0,1\} \)
- circuit **family** : a circuit for each input length \( C_1, C_2, C_3, \ldots = \{C_n\} \)
- \( \{C_n\} \) computes \( f \) iff for all \( x \)
  \[ C_{|x|}(x) = f(x) \]
- \( \{C_n\} \) decides \( L \), where \( L \) is the language associated with \( f \)

Connection to TMs

- given TM \( M \) running in time \( t(n) \) decides language \( L \)
- can build circuit family \( \{C_n\} \) that decides \( L \)
  - size of \( C_n = O(t(n)\^2) \)
  - Proof: CVAL construction
- Conclude: \( L \in P \) implies family of polynomial-size circuits that decides \( L \)

Connection to TMs

- other direction?

  - A poly-size circuit family:
    - \( C_n = (x \lor \neg x) \) if \( M_n \) halts
    - \( C_n = (x \land \neg x) \) if \( M_n \) loops
  - decides (unary version of) HALT!
  - oops…

Uniformity

- Strange aspect of circuit family:
  - can “encode” (potentially uncomputable) information in family specification
- solution: **uniformity** – require specification is simple to compute

  **Definition**: circuit family \( \{C_n\} \) is **logspace uniform** iff TM \( M \) outputs \( C_n \) on input \( 1^n \) and runs in \( O(\log n) \) space

Theorem: \( P = \) languages decidable by logspace uniform, polynomial-size circuit families \( \{C_n\} \).

- Proof:
  - already saw \( (\Rightarrow) \)
  - \( (\Leftarrow) \) on input \( x \), generate \( C_{|x|} \), evaluate it and accept iff output = 1
TMs that take advice

- family \{C_n\} without uniformity constraint is called "non-uniform"
- regard "non-uniformity" as a limited resource just like time, space, as follows:
  - add read-only "advice" tape to TM M
  - M "decides L with advice A(n)" iff
    \[ M(x, A(|x|)) \text{ accepts } \iff x \in L \]
  - note: A(n) depends only on |x|

Definition: \( \text{TIME}(t(n))/f(n) \) = the set of those languages L for which:
- there exists A(n) s.t. \(|A(n)| \leq f(n)\)
- TM M decides L with advice A(n) in time t(n)
- most important such class: \( P/poly = \bigcup_k \text{TIME}(n^k)/n^k \)

Theorem: \( L \in P/poly \) iff L decided by family of (non-uniform) polynomial size circuits.
- Proof:
  - (⇒) \( C_n \) from CVAL construction; hardwire advice A(n)
  - (⇐) define A(n) = description of \( C_n \); on input x, TM simulates \( C_{|x|}(x) \)

Approach to P/NP

- Believe \( \text{NP} \neq \text{P} \)
  - equivalent: "\text{NP} does not have uniform, polynomial-size circuits"
- Even believe \( \text{NP} \notin P/poly \)
  - equivalent: "\text{NP} (or e.g. SAT) does not have polynomial-size circuits"
  - implies \( \text{P} \neq \text{NP} \)
  - many believe: best hope for \( \text{P} \neq \text{NP} \)

Parallelism

- uniform circuits allow refinement of polynomial time:
  - circuit \( C \) depth \( \equiv \) parallel time
  - size \( \equiv \) parallel work

- the \( \text{NC} \) ("Nick’s Class") Hierarchy (of logspace uniform circuits):
  - \( \text{NC}_k = O(\log^k n) \) depth, poly(n) size
  - \( \text{NC} = \bigcup_k \text{NC}_k \)
  - captures "efficiently parallelizable problems"
  - not realistic? overly generous
  - OK for proving non-parallelizable
Matrix Multiplication

- what is the parallel complexity of this problem?
  - work = poly(n)
  - time = log^k(n)? (which k?)

n x n matrix A
n x n matrix B
n x n matrix AB

Matrix Multiplication

- two details
  - arithmetic matrix multiplication…
    A = (a_{i,k})  B = (b_{k,j})  (AB)_{i,j} = \sum_k (a_{i,k} \times b_{k,j})
  … vs. Boolean matrix multiplication:
    A = (a_{i,k})  B = (b_{k,j})  (AB)_{i,j} = \bigvee_k (a_{i,k} \wedge b_{k,j})

- single output bit: to make matrix multiplication a language: on input A, B, (i, j) output (AB)_{i,j}

Boolean formulas and \textbf{NC}_1

- Previous circuit is actually a formula. This is no accident:

\textbf{Theorem}: L \in \textbf{NC}_1 iff decidable by polynomial-size uniform* family of Boolean formulas.

* DSPACE(\log^k n)-uniform

Note: we measure formula size by leaf-size.

Boolean formulas and \textbf{NC}_1

- Proof:
  - (\Rightarrow) convert \textbf{NC}_1 circuit into formula
    - recursively:

- (\Leftarrow) convert formula of size n into formula of depth O(\log n)
  - note: size \leq 2^{O(\log^2 n)} so new formula has poly(n) size

key transformation
Boolean formulas and \( \text{NC}_1 \)

- D any minimal subtree with size at least \( n/3 \)
  - implies size(D) \( \leq 2n/3 \)
- define \( T(n) \) = maximum depth required for any size \( n \) formula
- \( \mathcal{C}_1, \mathcal{C}_0, D \) all size \( \leq 2n/3 \)

\[ T(n) \leq T(2n/3) + 3 \]

implies \( T(n) \leq O(\log n) \)

Relation to other classes

- Clearly \( \text{NC} \subseteq \text{P} \)
  - recall \( \text{P} \equiv \text{uniform poly-size circuits} \)
- \( \text{NC}_1 \subseteq \text{L} \)
  - on input \( x \), compose logspace algorithms for:
    - generating \( \mathcal{C}_0 \)
    - converting to formula
    - FVAL

\( \rightarrow \) \( \text{NL} \subseteq \text{NC}_2 \)

\( \text{S-T-CONN} \in \text{NC}_2 \)
- given \( G = (V, E) \), vertices \( s, t \)
- \( A = \) adjacency matrix (with self-loops)
- \((A^2)_{ij} = 1 \) iff path of length \( \leq 2 \) from node \( i \) to node \( j \)
- \((A^n)_{ij} = 1 \) iff path of length \( \leq n \) from node \( i \) to node \( j \)
- compute with depth \( \log n \) tree of Boolean matrix multiplications, output entry \( s, t \)
- \( \log^2 n \) depth total

\( \text{NC} \) vs. \( \text{P} \)

- can every efficient algorithm be efficiently parallelized?
  \( \text{NC} \neq \text{P} \)
- \( \text{P} \)-complete problems least-likely to be parallelizable
  - if \( \text{P} \)-complete problem is in \( \text{NC} \), then \( \text{P} = \text{NC} \)
  - Why? we use logspace reductions to show problem \( \text{P} \)-complete; \( \text{L} \) in \( \text{NC} \)

\( \text{NC}_1 \neq \text{P} \)

- can every uniform, poly-size Boolean circuit family be converted into a uniform, poly-size Boolean formula family?
  \( \text{NC}_1 = \text{P} \)