Complexity Theory

Lecture 11

Tom Gur

http://www.cl.cam.ac.uk/teaching/2324/Complexity

Configuration Graph

Define the *configuration graph* of M, x to be the graph whose nodes are the possible configurations, and there is an edge from i to j if, and only if, $i \rightarrow_M j$.

Configuration Graph

Define the *configuration graph* of M, x to be the graph whose nodes are the possible configurations, and there is an edge from i to j if, and only if, $i \rightarrow_M j$.

Then, M accepts x if, and only if, some accepting configuration is reachable from the starting configuration $(s, \triangleright, x, \triangleright, \varepsilon)$ in the configuration graph of M, x.

Using the $O(n^2)$ algorithm for Reachability, we get that $L(M)$—the language accepted by M—can be decided by a deterministic machine operating in time

$$c'(nc^{f(n)})^2 \sim c'c^{2(\log n + f(n))} \sim k^{(\log n + f(n))}$$

Using the $O(n^2)$ algorithm for Reachability, we get that $L(M)$—the language accepted by M—can be decided by a deterministic machine operating in time

$$c'(nc^{f(n)})^2 \sim c'c^{2(\log n + f(n))} \sim k^{(\log n + f(n))}$$

In particular, this establishes that NL $\subseteq$ P and NPSPACE $\subseteq$ EXP.

NL Reachability

We can construct an algorithm to show that the Reachability problem is in NL:

NL Reachability

We can construct an algorithm to show that the Reachability problem is in NL:

1. write the index of node a in the work space;

NL Reachability

We can construct an algorithm to show that the Reachability problem is in NL:

1. write the index of node *a* in the work space;
2. if *i* is the index currently written on the work space:

NL Reachability

We can construct an algorithm to show that the Reachability problem is in NL:

1. write the index of node a in the work space;
2. if i is the index currently written on the work space:
    2.1 if $i = b$ then accept, else
    guess an index j ($\log n$ bits) and write it on the work space.

NL Reachability

We can construct an algorithm to show that the Reachability problem is in NL:

1. write the index of node a in the work space;
2. if i is the index currently written on the work space:
   2.1 if $i = b$ then accept, else
   guess an index j ($\log n$ bits) and write it on the work space.
   2.2 if (i, j) is not an edge, reject, else replace i by j and return to (2).

Savitch's Theorem

Further simulation results for nondeterministic space are obtained by other algorithms for Reachability.

Savitch's Theorem

Further simulation results for nondeterministic space are obtained by other algorithms for Reachability.

We can show that Reachability can be solved by a *deterministic* algorithm in $O((\log n)^2)$ space.

Savitch's Theorem

Further simulation results for nondeterministic space are obtained by other algorithms for Reachability.

We can show that Reachability can be solved by a *deterministic* algorithm in $O((\log n)^2)$ space.

Consider the following recursive algorithm for determining whether there is a path from a to b of length at most i.

$O((\log n)^2)$ space Reachability algorithm:

$O((\log n)^2)$ space Reachability algorithm:

Path(a, b, i)

if $i = 1$ and $a \neq b$ and (a, b) is not an edge reject
else if (a, b) is an edge or $a = b$ accept
else, for each node x, check:

1. Path$(a, x, \lfloor i/2 \rfloor)$

if such an x is found, then accept, else reject.

$O((\log n)^2)$ space Reachability algorithm:

Path(a, b, i)

if $i = 1$ and $a \neq b$ and (a, b) is not an edge reject
else if (a, b) is an edge or $a = b$ accept
else, for each node x, check:

1. Path$(a, x, \lfloor i/2 \rfloor)$
2. Path$(x, b, \lceil i/2 \rceil)$

if such an x is found, then accept, else reject.

$O((\log n)^2)$ space Reachability algorithm:

Path(a, b, i)

if $i = 1$ and $a \neq b$ and (a, b) is not an edge reject
else if (a, b) is an edge or $a = b$ accept
else, for each node x, check:

1. Path$(a, x, \lfloor i/2 \rfloor)$
2. Path$(x, b, \lceil i/2 \rceil)$

if such an x is found, then accept, else reject.

$O((\log n)^2)$ space Reachability algorithm:

Path(a, b, i)

if $i = 1$ and $a \neq b$ and (a, b) is not an edge reject
else if (a, b) is an edge or $a = b$ accept
else, for each node x, check:

1. Path$(a, x, \lfloor i/2 \rfloor)$
2. Path$(x, b, \lceil i/2 \rceil)$

if such an x is found, then accept, else reject.

The maximum depth of recursion is $\log n$, and the number of bits of information kept at each stage is $3 \log n$.

Savitch's Theorem

The space efficient algorithm for reachability used on the configuration graph of a nondeterministic machine shows:

Savitch's Theorem

The space efficient algorithm for reachability used on the configuration graph of a nondeterministic machine shows:

$$\mathrm{NSPACE}(f) \subseteq \mathrm{SPACE}(f^2)$$

for $f(n) \geq \log n$.

Savitch's Theorem

The space efficient algorithm for reachability used on the configuration graph of a nondeterministic machine shows:

$$\mathrm{NSPACE}(f) \subseteq \mathrm{SPACE}(f^2)$$

for $f(n) \geq \log n$.

This yields

$$\mathrm{PSPACE} = \mathrm{NPSPACE} = \text{co-}\mathrm{NPSPACE}.$$

Complementation

A still more clever algorithm for Reachability has been used to show that nondeterministic space classes are closed under complementation:

Complementation

A still more clever algorithm for Reachability has been used to show that nondeterministic space classes are closed under complementation:

If $f(n) \geq \log n$, then

$$\mathrm{NSPACE}(f) = \text{co-}\mathrm{NSPACE}(f)$$

Complementation

A still more clever algorithm for Reachability has been used to show that nondeterministic space classes are closed under complementation:

If $f(n) \geq \log n$, then

$$\text{NSPACE}(f) = \text{co-NSPACE}(f)$$

In particular

$$\text{NL} = \text{co-NL}.$$

Logarithmic Space Reductions

We write

$$A \leq_L B$$

if there is a reduction f of A to B that is computable by a deterministic Turing machine using $O(\log n)$ workspace (with a *read-only* input tape and *write-only* output tape).

Logarithmic Space Reductions

We write

$$A \leq_L B$$

if there is a reduction f of A to B that is computable by a deterministic Turing machine using $O(\log n)$ workspace (with a *read-only* input tape and *write-only* output tape).

Note: We can compose $\leq_L$ reductions. So,

$$\text{if } A \leq_L B \text{ and } B \leq_L C \text{ then } A \leq_L C$$

NP-complete Problems

Analysing carefully the reductions we constructed in our proofs of NP-completeness, we can see that SAT and the various other NP-complete problems are actually complete under $\leq_L$ reductions.

NP-complete Problems

Analysing carefully the reductions we constructed in our proofs of NP-completeness, we can see that SAT and the various other NP-complete problems are actually complete under $\leq_L$ reductions.

Thus, if SAT $\leq_L A$ for some problem A in L then not only P $=$ NP but also L $=$ NP.

P-complete Problems

It makes little sense to talk of complete problems for the class P with respect to polynomial time reducibility $\leq_P$.

P-complete Problems

It makes little sense to talk of complete problems for the class P with respect to polynomial time reducibility $\leq_P$.

There are problems that are complete for P with respect to *logarithmic space* reductions $\leq_L$.

One example is CVP—the circuit value problem.

P-complete Problems

It makes little sense to talk of complete problems for the class P with respect to polynomial time reducibility $\leq_P$.

There are problems that are complete for P with respect to *logarithmic space* reductions $\leq_L$.

One example is CVP—the circuit value problem.

That is, for every language A in P,

$$A \leq_L \text{CVP}$$

- If $\text{CVP} \in \text{L}$ then $\text{L} = \text{P}$.

P-complete Problems

It makes little sense to talk of complete problems for the class P with respect to polynomial time reducibility $\leq_P$.

There are problems that are complete for P with respect to *logarithmic space* reductions $\leq_L$.

One example is CVP—the circuit value problem.

That is, for every language A in P,

$$A \leq_L \text{CVP}$$

- If $\text{CVP} \in \text{L}$ then $\text{L} = \text{P}$.
- If $\text{CVP} \in \text{NL}$ then $\text{NL} = \text{P}$.

Questions?