Overview

• DTM s and NTM s recognize the same languages.
• Resource bounded complexity classes.
• The classes \( P, NP, \) and \( EXP \).
  – \( P \subseteq NP \subseteq EXP \)
• Review of reductions between languages
  – From SAT to 3-SAT
• Definition of \( NP \)-complete languages
• The \( P \) versus \( NP \) question
• Introduction to a first \( NP \)-complete language
Nondeterministic TM (NTM)

- An NTM \((\Gamma, \Phi, Q, \delta, F, s)\) is a TM in which the FSM control unit has choice inputs from alphabet \(\Phi\).
- \(\delta: Q \times (\Gamma \times \Phi) \rightarrow Q \times \Gamma \times \{-1, 0, +1\}\) (halt on states in \(F\))

![Diagram of Nondeterministic TM](image)
Deterministic Simulation of an NTM

• Given an NTM N recognizing L, can we construct a DTM D that also recognizes L?
  — Yes.

• Approach:
  — Assume that N accepts strings $w$ in $L$, $|w| = n$, in $T(n)$ steps. Construct DTM D that tries all choice inputs of length $\leq T(n)$ until D accepts or all choices exhausted, in which case, we reject.
Three-Tape DTM D Simulating NTM N

1. Input $w$ on read-only tape
2. Copy $w$ to work tape
3. Place a choice input string on enumeration tape (Choices: 0, 1, 00, 01, 10, 11, 000, ...)
4. If current choice doesn’t work, try the next one.
5. Read 1 choice per step and simulate $N$ on it, until choices exhausted or $w$ accepted.
6. If exhausted, erase work tape and go to 2.
DTM D Simulating NTM N

• The above DTM D accepts every \( w \) in \( L \) in \( p(|w|) \) steps. If \( w \) not in \( L \), it never halts.

• A DTM \( D_H \) can be constructed from D that halts in \( O(2^{p(|w|)}p(|w|)) \) steps.
  – \( D_H \) puts \( p(|w|) + 1 \) special symbols, say \#\, on blank enumeration tape. \( D_H \) can tell if the last choice sequence of length \( p(|w|) \) has been written on the tape because exactly one instance of \# will remain.
  – There are \( 2^{p(|w|)+1}-1 \) binary strings of length \( p(|w|) \).
Nondeterminism Doesn’t Help

• It follows that nondeterminism doesn’t increase the set of languages recognized by Turing machines.

• However, it may effect the running time of a Turing machine to recognize a language.
Resource-Bounded Language Recognition

• Languages are classified by the amount of time or space needed to recognize them on a TM.
• Always measured in terms of input length.

• \( \textbf{P} \) and \( \textbf{NP} \) are languages recognized in polynomial time in length of input on
  – \( \textbf{P} \): deterministic Turing machines and
  – \( \textbf{NP} \): nondeterministic Turing machines.
Recall The Class $P$

- **Definition** A language $L \subseteq \Gamma^*$ is in $P$ if there is a TM $M_L$ with tape alphabet $\Gamma$ and polynomial $p(n)$ such that for every $w$ in $\Gamma^*$,
  - $M_L$ halts in $p(|w|)$ steps and
  - $M_L$ accepts $w$ if and only if $w$ is in $L$. 

© John E. Savage
The Class \textbf{NP}

• A language \( L \subseteq \Gamma^* \) is in \textbf{NP} if there is an NTM \( M_L \) and a polynomial \( p(n) \) such that
  – \( M_L \) halts and accepts each \( w \) in \( L \) with the aid of the choice agent in \( p(|w|) \) steps.
  – If \( w \) is not in \( L \), \( M_L \) does not accept \( w \).
    • \( M_L \) does not accept either by halting or looping.
The Class **EXP**

- A language $L \subseteq \Gamma^*$ is in **EXP** if there is a DTM $M_L$ and a exponential function $e(n)$ such that
  - for every $x \in \Gamma^*$, $M_L$ halts in $e(|x|)$ steps and
  - $M_L$ accepts $x$ if it is in $L$ and rejects it otherwise.

- Colloquially, if $L$ is in **EXP**, there exists a DTM that halts on all inputs in an exponential number of steps that accepts only strings in $L$. 
Class Inclusions

• Previously we showed that $P \subseteq NP$.
  – Because a DTM is a restricted form of NTM, every language recognized by a TM in $P$ is also in $NP$.

• The simulation given above of an NTM by DTM shows that $NP \subseteq EXP$.

• It follows that $P \subseteq NP \subseteq EXP$.
The Role of Reductions

• If problem A is hard to solve and we can reduce it efficiently to problem B, is B easy to solve?
  – If B has an efficient solution, let’s solve hard problem A by reducing it efficiently to B and then solving B.
  – This provides an efficient algorithm for A.
  – But since A is hard, B can’t have an efficient solution.

• Thus, if we can show that A is hard, using efficient reductions other problems can be shown hard.

© John E. Savage
The Role of Reductions

• Reductions are used to identify problems that are
  – $\textbf{NP}$-complete – hard to compute efficiently on serial computers.
  – $\textbf{P}$-complete – hard to parallelize efficiently
  – Impossible – e.g. the Halting Problem

• The Halting Problem – an impossible problem
  – No TM algorithm exists that can tell if another TM will halt on its input or not.
  – Once we show Halting Problem is impossible, we can do the same for other problems through reductions.
Formal Definition of a Reduction

- **Definition:** A reduction from language $L_1 \subseteq \Gamma^*$ to language $L_2 \subseteq \Sigma^*$ is a function $f : \Gamma^* \to \Sigma^*$ computable by a DTM such that $w \in L_1$ if and only if $w' = f(w) \in L_2$. ($f$ translates $L_1$ to $L_2$.)

![Diagram showing a reduction process]

This Recognizer for $L_1$ invokes the recognizer $R$ for $L_2$. © John E. Savage
Examples of Reductions

• Simple reduction:
  – $L_a = \{ w \mid w \in \{a,b\}^* \text{ has odd number of a’s} \}$
  – $L_b = \{ w \mid w \in \{0,1\}^* \text{ has odd number of 1’s} \}$
  – Let $f(a) = 1$ and $f(b) = 0$.

• More complex reduction:
  – From SAT to 3-SAT.
Satisfiability (SAT)

• **Instance**: A set of literals \( X = \{\tilde{x}_1, \ldots, \tilde{x}_n\} \) and clauses \( C = (c_1, \ldots, c_n) \), each \( c_i \) is a subset of \( X \).

• “Yes” **Instance**: There exists an assignment to variables over \( \{0,1\} \) such that each clause has a literal with value 1.

• **Example**:
  - “Yes” instance: \((x_1 + x_2) (\bar{x}_1 + x_2) (\bar{x}_1 + \bar{x}_2)\)
  - “No” instance: \((x_1 + x_2) (\bar{x}_1 + x_2) (\bar{x}_1 + \bar{x}_2) (x_1 + \bar{x}_2)\)

  **Note**: We let + denote OR.
3-SAT

• **Instance**: Set of literals $X = \{\tilde{x}_1, ..., \tilde{x}_n\}$ & clauses $C = (c_1, ..., c_n)$, each $c_i$ is a subset of $X$, $|c_i| \leq 3$.
• “Yes” **Instance**: There exists an assignment to variables over $\{0,1\}$ such that each clause has a literal with value 1.
  – Example: $(x_1 + \tilde{x}_2 + x_3) (\tilde{x}_1 + x_3 + x_4) (x_2 + \tilde{x}_4)$
Reduction from SAT to 3-SAT

• If $n > 3$ replace $(y_1+y_2+...+y_n)$ by a set of clauses in at most three literals by introducing new variables.
  – E.g. Replace $(y_1+y_2+y_3+y_4)$ by $(y_1+y_2+z_1) (\bar{z}_1+y_3+y_4)$.

• Proof is by induction using example. Assume that it applies to $n$ for $n > 3$. Show works for $n+1$. Replace $(y_1+y_2+...+y_n+y_{n+1})$ by $(y_1+...y_{n-1}+z_1) (\bar{z}_1+y_n+y_{n+1})$ and apply inductive assumption on the first term since it has $n$ variables.
Reduction from SAT to 3-SAT (cont.)

• Is the following equivalent to \((y_1+y_2+...+y_n)\)?
  \((y_1+y_2+z_1) (\bar{z}_1+y_3+z_2)(\bar{z}_2+y_4+z_3) \ldots (\bar{z}_{n-3}+y_{n-1}+y_n)\)

• \((y_1+y_2+...+y_n)\) is satisfied iff \(y_j = 1\) for some \(j\).

• If \(y_1+y_2 = 1\), let \(z_j = 0\) for \(j \geq 1\). If \(y_r = 0\) for \(r \leq k\) when \(k \geq 2\) and \(y_k = 1\), let \(z_j = 0\) for \(j \leq k-2\) and \(z_j = 0\) for \(j \geq k-1\). Then, \((y_1+y_2+...+y_n)\) is satisfied iff the remaining \(n-2\) clauses are satisfied.
Reduction from SAT to 3-SAT

• This is a computable reduction.

• Is it computable in polynomial time in the length of an instance on a DTM?

• A clause with \( t \) literals can be expanded in time \( O(t) \) on a RAM. Thus, if the number of literals is \( T \), expansion can be done in \( O(T) \) time on a RAM and \( O(T^2) \) time on DTM.
Polynomial-Time Reduction

- **Definition:** A polynomial-time reduction (P-time) from language $L_1 \subseteq \Gamma^*$ to language $L_2 \subseteq \Sigma^*$ is a reduction $f : \Gamma^* \rightarrow \Sigma^*$ computable by a DTM in time polynomial in the length of its input. ($f$ P-time translates $L_1$ to $L_2$.)
NP-Complete Language

• A language $L \subseteq \Gamma^*$ is **NP-complete** if
  – it is in **NP** and
  – for every language $L_0 \subseteq \Gamma^*$ in **NP**, there is a P-time reduction $f_0$ from $L_0$ to $L$.

• If only second condition holds, we say $L$ is **NP-hard**.

Recognize for $L_0$ in **NP** invokes recognizer $R$ for $L$.

Recognizer $R$ for $L$
Is $P = NP$?

- Because an $NP$-complete language is in $NP$, it can be $P$-time reduced to another such language.
- Thus, an $NP$-complete language is a “hardest” language in $NP$ within polynomial bounds.

- If one $NP$-complete language recognizer requires exponential time, all do and $P \neq NP$.
  - Recall definition of a reduction.
- If one $NP$-complete language is in $P$, all are in $P$. That is, $P = NP$. 

© John E. Savage

CSCI 1010 Lect 9
Circuit Satisfiability

- A one-output circuit is **satisfiable** if its inputs can be chosen so that the output is 1.
- We generalize as follows:
  - A circuit is **satisfiable** if for fixed values of its deterministic inputs its nondeterministic inputs can be chosen so that the output is 1.
  - Previously all inputs were nondeterministic.

- CIRCUIT SAT is the set of satisfiable circuits.
A First \textbf{NP}-Complete Problem

• To show that language \( L \) is \textbf{NP}-complete we must show that it is in \textbf{NP} and every language in \textbf{NP} can be reduced to \( L \) in P-time.

• Our first \textbf{NP}-complete language is CIRCUIT SAT.

• Next time we give a P-time algorithm \( A \) that given an NTM and a polynomial time bound \( p(n) \) for an arbitrary language \( L_0 \) in \textbf{NP}, \( A \) prints an instance of a circuit that is satisfiable on input \( w \) iff \( w \in L_0 \). The “Yes” circuit instances of these circuits form CIRCUIT SAT.
Simulating NTM with a Circuit

• Next time we design circuit to simulate the tape and control units with circuits.
• The combined circuit is satisfiable iff on input $w$ the NTM can accept the input.
Review

- DTM$s$ and NTM$s$ recognize the same languages.
- Resource bounded complexity classes.
- The classes $P$, $NP$, and $EXP$.
  - $P \subseteq NP \subseteq EXP$
- Review of reductions between languages
  - From SAT to 3-SAT
- Definition of $NP$-complete languages
- The $P$ versus $NP$ question
- Introduction to a first $NP$-complete language