CSCI 1010
Models of Computation

Lecture 07
Nondeterministic Computation
Overview

• Functions computed by Boolean functions.
• Nondeterministic circuits and language recognition.
• Circuit SAT – a first \textbf{NP}-complete language.
• Nondeterministic FSMs
  – Language recognition
• Deterministic & nondeterministic TM
  – Language recognition
• Nondeterministic TM for \textbf{Circuit SAT}
Functions Computed by Circuits

• A **Boolean function**
  \[ f : \{0,1\}^n \rightarrow \{0,1\} \]

• A **binary function**
  \[ f : \{0,1\}^n \rightarrow \{0,1\}^m \]

• A logic circuit computes a binary function.

• Here \((c,s)\) represents \(x+y+z\) in binary notation.
  \[(c,s) = f(x,y,z).\]

\(\oplus\) is **Exclusive OR**
Language Recognition by Circuits

• Let $f : \{0,1\}^n \rightarrow \{0,1\}^m$ be binary function. Each output can be associated with two languages.
• Let $g : \{0,1\}^n \rightarrow \{0,1\}$ be one of these functions. The inputs $u$ for which $g(u) = 1$ forms one language. Those where $g(u) = 0$ forms another.

• In the example, $c = g(x,y,z) = 1$ when $x+y+z \geq 2$ and $c = 0$ when $x+y+z < 2$.
• $s = h(x,y,z) = 1$ when $x \oplus y \oplus z = 1$ where $\oplus$ is Exclusive OR and $s = 0$ when $x \oplus y \oplus z = 0$. 
Nondeterministic Circuit Computations

• Until now all circuit inputs are deterministic (D), that is, fixed in advance of a computation.
• Some problems are more easily defined using nondeterministic (ND) inputs, that is, inputs that are not fixed in advance of a computation.
• ND computations are done with Boolean (one-output) functions.
ND Circuit Computations

• Let $C$ be a circuit with variables in $X = \{x_1, \ldots, x_n\}$ computing function $f(x_1, \ldots, x_n), f : \{0,1\}^n \rightarrow \{0,1\}$

• Let $X_D, X_{ND} \subseteq X, X_D \cap X_{ND} = \phi, |X_D| = k, |X_{ND}| = n-k$

• Then $C$ **ND-computes** function $f^* : \{0,1\}^k \rightarrow \{0,1\}$ for $X_D \subseteq X$ where $f^*(X_D) = 1$ if there are $n-k$ values for $X_{ND}$ such that $f(X_D, X_{ND})=1$. Else $f^*(X_D)=0$
  
  – If possible, a **helper** chooses $X_{ND}$ so that $f = 1$. 
Functions Computed by ND Circuits

• Then \( C_{ND} \) computes function \( f^*:\{0,1\}^k \rightarrow \{0,1\} \) where for \( X_D \subseteq X, f^*(X_D) = 1 \) if there is an \( n \)-tuple \( X_{ND} \) s.t. \( f(X_D,X_{ND}) = 1 \). Else \( f^*(X_D) = 0 \).
  – If it can, a helper chooses \( X_{ND} \) so that \( f = 1 \).
Example

• **Example:** Let \( c = f(x,y,z) = 1 \) when \( x+y+z \geq 2 \).
• Let \( X_D = \{x,y\} \) and \( X_{ND} = \{z\} \). Then, \( f^* = x \lor y \) (OR)
• When \( x=y=0 \), \( c = 0 \) for all \( z \).
• When either \( x \) or \( y = 1 \), let \( z = 1 \).
Circuit Satisfiability (Circuit SAT)

• A one-output circuit is **satisfiable** if its free inputs can be chosen so that the output is 1.
  – Fixed inputs are D. Free inputs are in ND.

• One-output circuits (SLPs) are either a) satisfiable or b) not satisfiable.

• **Circuit SAT** is set of satisfiable circuits of all lengths
  – Can we recognize Circuit SAT (set of satisfying SLPs) in polynomial time, that is, in time \(O(|SLP|^k)\) for some \(k\)?
  – That is, can we build a TM that recognizes Circuit SAT in polynomial time? (It decides SLP membership in P-time.)
Circuit SAT

- Circuit SAT is set of satisfiable circuits.
  - Not thought possible to tell if an SLP is satisfiable in polynomial time, i.e. $O(|SLP|^k)$ for some $k$.
  - Why is that?

- We can **verify** SLP membership in $O(|SLP|)$ time!
  - Given satisfying values for $X_{ND}$, $f$ can be evaluated by a TM in time that is polynomial in the size of the SLP for $f$.

- Later we show that Circuit SAT is an **NP**-complete language!

- These languages are “really hard” to recognize.
Circuit SAT vs ND Circuits

• **Yes Instances** of Circuit SAT: The set of circuits (SLPs) that *do* have satisfying inputs.

• **No Instances** of Circuit SAT: The set of circuits (SLPs) that have *no* satisfying inputs.

• **Note:** Yes instances nondeterministically compute the Boolean function $f^* = 1$. 
Deterministic FSM (DFSM)

- A DFSM recognizer \((\Sigma, Q, \delta, F, s)\), has states \(Q\), input alphabet \(\Sigma\), next state function \(\delta:Q \times \Sigma \rightarrow Q\), start state \(s\), and accept states \(F\).
- It accepts input strings that cause it to end in a final state in \(F\). It recognizes the language of these strings.
Nondeterministic FSMs (NFSMs)

• An NFSM recognizer is a DFSM recognizer that has two inputs, user \((\Sigma)\) and choice \((\Phi)\) inputs.
• Choice inputs play role of ND inputs in circuits.
• User can choose to move to more than 1 state.
Language Recognition by NFSMs

• An NFSM recognizer \((\Sigma, \Phi, Q, \delta, F, s)\) has states \(Q\), user input alphabet \(\Sigma\), choice input alphabet \(\Phi\), next state function \(\delta: Q \times \Sigma \times \Phi \rightarrow Q\), start state \(s\), and accept states \(F\).

• A sequence of \(n\) user inputs (in \(\Sigma\)) is accepted if \(n\) choice inputs (in \(\Phi\)) exist that cause the NFSM to enter an accept state.

• The language recognized by an NFSM is the set of accepted user inputs.
Role of NFSMs

• Later we will ask if NFSMs recognize languages that cannot be recognized by DFSMs.
  – The answer will be “No”.

• We introduce nondeterminism for FSMs because it helps to understand nondeterministic Turing machines.
Example of an NFSM

• Choice inputs are in parentheses.
• This NFSM accepts strings ending in 00101.

If next state doesn’t depend on choice Input, no parenthesis after input. Some moves are not allowed!

• This one accepts all binary strings.
Another Example

- **DFSM & NFSM accepting** \( \{0,1\}^*101 \). Try 1101 & 1001.

![Finite State Machine Diagram]
Role of Nondeterminism

• Nondeterminism makes it easy to define the class of \textbf{NP}-complete problems.

• Recall: Circuit SAT is set of non-deterministic one-output circuits computing function $f^* = 1$.
  – The circuits are satisfiable.
Deterministic TM (DTM)

- A DTM \((Q, \Sigma, \delta, F, s)\) has **states** \(Q\), **tape alphabet** \(\Sigma\) (no blanks), **output pairs** \((\Sigma \cup \{\beta\}) \times \{+1, 0, -1\}\) and (the control unit) **start state** \(s\).
Deterministic TM (DTM)

- A DTM $(Q, \Sigma, \delta, F, s)$ has **states** $Q$, **tape alphabet** $\Sigma$ (no blanks), **output pairs** $(\Sigma \cup \{\beta\}) \times \{+1, 0, -1\}$ and (the control unit) **start state** $s$.

- **Next state function** $\delta: Q \times \Sigma \to Q \times (\Sigma \cup \{\beta\}) \times \{+1, 0, -1\}$ maps the current state and tape cell value to a new state, new tape cell value, and head movement ($+1 = \text{right}$, $-1 = \text{left}$, $0 = \text{no move}$).
How to Design Turing Machines

• Except for efficiency, a TM is like the random access machine (RAM) programming model.

• Section 3.8 of Models of Computation simulates RAM by TM & vice versa.

• To design (or program) a TM, first construct a RAM algorithm for the problem. Then, translate it into a TM program.
Nondeterministic TM (NTM)

• An NTM is a DTM \((Q, \Sigma \times \Phi, \delta, F, s)\) where 
  \(\delta: Q \times \Sigma \times \Phi \rightarrow Q \times (\Sigma \cup \{\beta\}) \times \{+1, 0, -1\}\) and the DTM tape alphabet \(\Sigma\) is replaced by a new alphabet \(\Sigma \times \Phi\) where \(\Phi\) is the choice alphabet.
Nondeterministic TM (NTM)

• An NTM is a DTM \((Q, \Sigma \times \Phi, \delta, F, s)\) where 
  \(\delta: Q \times \Sigma \times \Phi \rightarrow Q \times (\Sigma \cup \{\beta\}) \times \{+1, 0, -1\}\) and the DTM tape alphabet \(\Sigma\) is replaced by a new alphabet \(\Sigma \times \Phi\).

• \(\Sigma\) is the tape alphabet & \(\Phi\) is the choice alphabet. When both symbols are known, the NTM acts like a DTM.

• An NTM **accepts** a string \(x\) placed left-adjusted on its otherwise blank tape if there are values for choice inputs that cause the NTM to enter an accept state.

• An NTM **recognizes** the language of strings it accepts.
Justification for Nondeterminism

• Choice inputs simplify language definitions, as we shall see.
NTM for Circuit SAT

• We sketch an NTM that halts and accepts only Yes instances of Circuit SAT.

• An instance of Circuit SAT (an SLP) is written on the tape of the NTM. CU uses one choice bit for each circuit input and writes them on the tape.

• It then operates as a DTM and computes the value of the circuit on the choice inputs. It accepts only if the result is 1 (True).

• This machine verifies its choices quickly (p-time)
The Role of the NTM

• The NTM is a powerful means to define important classes of languages.

• We use it to define \( \text{NP} \)-complete languages.

• We will ask questions such as “Does an NTM recognize languages in polynomial time that a DTM cannot?”.
A Taste of \textbf{NP}-Complete Languages

- \textbf{NP} is the class of languages that can be recognized in non-deterministic polynomial time.
- An \textbf{NP}-complete language \(L\) is in \textbf{NP} if recognition of any language in \textbf{NP} can be reduced to recognition of \(L\) in deterministic p-time.
- If one \textbf{NP}-complete language can be recognized in deterministic polynomial time, they all can.
- Unknown whether \textbf{NP}-complete languages are in poly time (\(\mathcal{P}\)) or not.
Review

- Functions computed by Boolean functions.
- Nondeterministic circuits and language recognition.
- Circuit SAT – a first \textbf{NP}-complete language.
- Nondeterministic FSMs
  - Language recognition
- Deterministic & nondeterministic TM
  - Language recognition
- Nondeterministic TM for \textbf{Circuit SAT}
- Intro to \textbf{NP}-complete languages