Circuit satisfiability and
Cook-Levin Theorem

Lecture 25
Thursday, April 25, 2019
Recap

**NP**: languages that have non-deterministic polynomial time algorithms
**Recap**

**NP**: languages that have non-deterministic polynomial time algorithms

A language $L$ is **NP-Complete** iff

- $L$ is in **NP**
- for every $L'$ in **NP**, $L' \leq_P L$
Recap

**NP**: languages that have non-deterministic polynomial time algorithms

A language $L$ is **NP-Complete** iff

- $L$ is in **NP**
- for every $L'$ in **NP**, $L' \leq_P L$

$L$ is **NP-Hard** if for every $L'$ in **NP**, $L' \leq_P L$. 

Theorem (Cook-Levin)

SAT is NP-Complete.
Recap

**NP**: languages that have non-deterministic polynomial time algorithms

A language $L$ is **NP-Complete** iff

- $L$ is in **NP**
- for every $L'$ in **NP**, $L' \leq_P L$

$L$ is **NP-Hard** if for every $L'$ in **NP**, $L' \leq_P L$.

**Theorem (Cook-Levin)**

**SAT** is **NP-Complete**.
Pictorial View

\[ \text{P} \subseteq \text{NP} \subseteq \text{NP-C} \subseteq \text{NP-Hard} \]
Possible scenarios:

1. $P = NP$.
2. $P \neq NP$
Possible scenarios:

1. $P = NP$.
2. $P \neq NP$

Question: Suppose $P \neq NP$. Is every problem in $NP \setminus P$ also NP-Complete?
Possible scenarios:

1. \( P = NP \).
2. \( P \neq NP \)

Question: Suppose \( P \neq NP \). Is every problem in \( NP \setminus P \) also NP-Complete?

**Theorem (Ladner)**

*If \( P \neq NP \) then there is a problem/language \( X \in NP \setminus P \) such that \( X \) is not NP-Complete.*
NP-Complete Problems

Previous lectures:

- 3-SAT
- Independent Set
- Hamiltonian Cycle
- 3-Color

Today:

- Circuit SAT
- SAT

Important: understanding the problems and that they are hard.

Proofs and reductions will be sketchy and mainly to give a flavor
Part I

Circuit SAT
Figure 10.1. An AND gate, an OR gate, and a NOT gate.

Figure 10.2. A boolean circuit. Inputs enter from the left, and the output leaves to the right.
Circuits

Definition

A circuit is a directed *acyclic* graph with

1. **Input** vertices (without incoming edges) labelled with 0, 1 or a distinct variable.
2. Every other vertex is labelled \(\lor, \land\) or \(\neg\).
3. Single node **output** vertex with no outgoing edges.
**CSAT**: Circuit Satisfaction

**Definition (Circuit Satisfaction (**CSAT**).)**

Given a circuit as input, is there an assignment to the input variables that causes the output to get value 1?

**Claim**: **CSAT** is in **NP**.

**Certificate**: Assignment to input variables.

**Certifier**: Evaluate the value of each gate in a topological sort of DAG and check the output gate value.
**CSAT**: Circuit Satisfaction

**Definition (Circuit Satisfaction (CSAT).)**
Given a circuit as input, is there an assignment to the input variables that causes the output to get value 1?

**Claim**

**CSAT** is in **NP**.

1. **Certificate**: Assignment to input variables.

2. **Certifier**: Evaluate the value of each gate in a topological sort of **DAG** and check the output gate value.
Circuit SAT vs SAT

CNF formulas are a rather restricted form of Boolean formulas.

Circuits are a much more powerful (and hence easier) way to express Boolean formulas

\[(\neg x_4 \lor x_2 \lor x_3) \land (x_2 \lor \neg x_1 \lor x_4)\]
Circuit SAT vs SAT

**CNF** formulas are a rather restricted form of Boolean formulas.

Circuits are a much more powerful (and hence easier) way to express Boolean formulas.

However they are equivalent in terms of polynomial-time solvability.

**Theorem**

\[ \text{SAT} \leq_p \text{3SAT} \leq_p \text{CSAT}. \]

**Theorem**

\[ \text{CSAT} \leq_p \text{SAT} \leq_p \text{3SAT}. \]
Converting a **CNF** formula into a Circuit

**3SAT \( \leq_p \) CSAT**

Given **3CNF** formula \( \varphi \) with \( n \) variables and \( m \) clauses, create a Circuit \( C \).

- Inputs to \( C \) are the \( n \) boolean variables \( x_1, x_2, \ldots, x_n \)
- Use NOT gate to generate literal \( \neg x_i \) for each variable \( x_i \)
- For each clause \( (\ell_1 \lor \ell_2 \lor \ell_3) \) use two OR gates to mimic formula
- Combine the outputs for the clauses using AND gates to obtain the final output
Example

$3\text{SAT} \leq_p \text{CSAT}$

$$\varphi = \left( x_1 \lor x_3 \lor x_4 \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor x_4 \right)$$
The other direction: $\text{CSAT} \leq_p 3\text{SAT}$

1. Now: $\text{CSAT} \leq_p \text{SAT}$
Converting a circuit into a **CNF** formula

**Label the nodes**

(A) Input circuit

(B) Label the nodes.
Converting a circuit into a **CNF** formula

Introduce a variable for each node

(B) Label the nodes.

(C) Introduce var for each node.
Converting a circuit into a **CNF** formula

Write a sub-formula for each variable that is true if the var is computed correctly.

(C) Introduce var for each node.
(D) Write a sub-formula for each variable that is true if the var is computed correctly.

\[
\begin{align*}
x_k \quad \text{(Demand a sat’ assignment!)} \\
x_k &= x_i \land x_j \\
x_j &= x_g \land x_h \\
x_i &= \neg x_f \\
x_h &= x_d \lor x_e \\
x_g &= x_b \lor x_c \\
x_f &= x_a \land x_b \\
x_d &= 0 \\
x_e &= 1 \\
x_a &= 1
\end{align*}
\]
Reduction: $\text{CSAT} \leq \text{P SAT}$

1. For each gate (vertex) $v$ in the circuit, create a variable $x_v$

2. Case $\neg$: $v$ is labeled $\neg$ and has one incoming edge from $u$ (so $x_v = \neg x_u$). In SAT formula generate, add clauses $(x_u \lor x_v)$, $(\neg x_u \lor \neg x_v)$. Observe that

$$x_v = \neg x_u \text{ is true } \iff (x_u \lor x_v) \land (\neg x_u \lor \neg x_v) \text{ both true.}$$
Reduction: \( \text{CSAT} \leq_{P} \text{SAT} \)

Continued...

1. **Case \( \lor \):** So \( x_v = x_u \lor x_w \). In \( \text{SAT} \) formula generated, add clauses \((x_v \lor \neg x_u), (x_v \lor \neg x_w), \) and \((\neg x_v \lor x_u \lor x_w)\). Again, observe that

\[
(x_v = x_u \lor x_w) \text{ is true } \iff \begin{cases} 
(x_v \lor \neg x_u), \\
(x_v \lor \neg x_w), \\
(\neg x_v \lor x_u \lor x_w) \end{cases} \text{ all true.}
\]
Reduction: $\text{CSAT} \leq_P \text{SAT}$

Continued...

**Case $\land$:** So $x_v = x_u \land x_w$. In $\text{SAT}$ formula generated, add clauses $(\neg x_v \lor x_u)$, $(\neg x_v \lor x_w)$, and $(x_v \lor \neg x_u \lor \neg x_w)$. Again observe that

$x_v = x_u \land x_w$ is true $\iff (\neg x_v \lor x_u), (\neg x_v \lor x_w), (x_v \lor \neg x_u \lor \neg x_w)$ all true.
1. If \( v \) is an input gate with a fixed value then we do the following. If \( x_v = 1 \) add clause \( x_v \). If \( x_v = 0 \) add clause \( \neg x_v \).

2. Add the clause \( x_v \) where \( v \) is the variable for the output gate.
Converting a circuit into a **CNF** formula

Convert each sub-formula to an equivalent CNF formula

<table>
<thead>
<tr>
<th>$x_k$</th>
<th>$x_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_k = x_i \land x_j$</td>
<td>$(\neg x_k \lor x_i) \land (\neg x_k \lor x_j) \land (x_k \lor \neg x_i \lor \neg x_j)$</td>
</tr>
<tr>
<td>$x_j = x_g \land x_h$</td>
<td>$(\neg x_j \lor x_g) \land (\neg x_j \lor x_h) \land (x_j \lor \neg x_g \lor \neg x_h)$</td>
</tr>
<tr>
<td>$x_i = \neg x_f$</td>
<td>$(x_i \lor x_f) \land (\neg x_i \lor \neg x_f)$</td>
</tr>
<tr>
<td>$x_h = x_d \lor x_e$</td>
<td>$(x_h \lor \neg x_d) \land (x_h \lor \neg x_e) \land (\neg x_h \lor x_d \lor x_e)$</td>
</tr>
<tr>
<td>$x_g = x_b \lor x_c$</td>
<td>$(x_g \lor \neg x_b) \land (x_g \lor \neg x_c) \land (\neg x_g \lor x_b \lor x_c)$</td>
</tr>
<tr>
<td>$x_f = x_a \land x_b$</td>
<td>$(\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (x_f \lor \neg x_a \lor \neg x_b)$</td>
</tr>
<tr>
<td>$x_d = 0$</td>
<td>$\neg x_d$</td>
</tr>
<tr>
<td>$x_a = 1$</td>
<td>$x_a$</td>
</tr>
</tbody>
</table>
We got a **CNF** formula that is satisfiable if and only if the original circuit is satisfiable.

\[
x_k \land (\neg x_k \lor x_i) \land (\neg x_k \lor x_j) \land (x_k \lor \neg x_i \lor \neg x_j) \land (\neg x_j \lor x_g) \land (\neg x_j \lor x_h) \land (x_j \lor \neg x_g \lor \neg x_h) \land (x_i \lor x_f) \land (\neg x_i \lor \neg x_f) \land (x_i \lor x_h) \land (\neg x_i \lor \neg x_f) \land (\neg x_j \lor x_i) \land (\neg x_j \lor \neg x_g) \land (\neg x_h \lor x_d) \land (\neg x_h \lor \neg x_e) \land (x_h \lor \neg x_d) \land (x_h \lor \neg x_e) \land (\neg x_h \lor x_d \lor x_e) \land (x_h \lor \neg x_f) \land (\neg x_h \lor \neg x_e) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g) \land (\neg x_f \lor x_c) \land (\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (\neg x_f \lor x_g)
\]
Correctness of Reduction

Need to show circuit \( C \) is satisfiable iff \( \varphi_C \) is satisfiable

\[ \Rightarrow \] Consider a satisfying assignment \( a \) for \( C \)

1. Find values of all gates in \( C \) under \( a \)
2. Give value of gate \( v \) to variable \( x_v \); call this assignment \( a' \)
3. \( a' \) satisfies \( \varphi_C \) (exercise)

\[ \Leftarrow \] Consider a satisfying assignment \( a \) for \( \varphi_C \)

1. Let \( a' \) be the restriction of \( a \) to only the input variables
2. Value of gate \( v \) under \( a' \) is the same as value of \( x_v \) in \( a \)
3. Thus, \( a' \) satisfies \( C \)
Part II

Proof of Cook-Levin Theorem
Cook-Levin Theorem

Theorem (Cook-Levin)

**SAT** is **NP-Complete**.

We have already seen that **SAT** is in **NP**.

Need to prove that every language $L \in \text{NP}$, $L \leq_P \text{SAT}$.
Cook-Levin Theorem

Theorem (Cook-Levin)

**SAT** is **NP-Complete**.

We have already seen that **SAT** is in **NP**.

Need to prove that every language \( L \in \text{NP}, L \leq_P \text{SAT} \)

**Difficulty**: Infinite number of languages in **NP**. Must simultaneously show a *generic* reduction strategy.
High-level Plan

What does it mean that $L \in \text{NP}$?

$L \in \text{NP}$ implies that there is a non-deterministic TM $M$ and polynomial $p()$ such that

$$L = \{x \in \Sigma^* \mid M \text{ accepts } x \text{ in at most } p(|x|) \text{ steps} \}$$
What does it mean that \( L \in \text{NP} \)?

\( L \in \text{NP} \) implies that there is a non-deterministic TM \( M \) and polynomial \( p() \) such that

\[
L = \{ x \in \Sigma^* \mid M \text{ accepts } x \text{ in at most } p(|x|) \text{ steps} \}
\]

We will describe a reduction \( f_M \) that depends on \( M, p \) such that:

- \( f_M \) takes as input a string \( x \) and outputs a SAT formula \( f_M(x) \)
- \( f_M \) runs in time polynomial in \( |x| \)
- \( x \in L \) if and only if \( f_M(x) \) is satisfiable
$f_M(x)$ is satisfiable if and only if $x \in L$

$f_M(x)$ is satisfiable if and only if nondeterministic $M$ accepts $x$ in $p(|x|)$ steps
Plan continued

\(f_M(x)\) is satisfiable if and only if \(x \in L\)

\(f_M(x)\) is satisfiable if and only if nondeterministic \(M\) accepts \(x\) in \(p(|x|)\) steps

BIG IDEA

- \(f_M(x)\) will express “\(M\) on input \(x\) accepts in \(p(|x|)\) steps”
- \(f_M(x)\) will encode a computation history of \(M\) on \(x\)
- \(f_M(x)\) will be a carefully constructed CNF formula s.t if we have a satisfying assignment to it, then we will be able to see a complete accepting computation of \(M\) on \(x\) down to the last detail of where the head is, what transition is chosen, what the tape contents are, at each step.
Tableau of Computation

\( M \) runs in time \( p(|x|) \) on \( x \). Entire computation of \( M \) on \( x \) can be represented by a “tableau”

Row \( i \) gives contents of all cells at time \( i \)
At time 0 tape has input \( x \) followed by blanks
Each row long enough to hold all cells \( M \) might ever have scanned.
Variables of $f_M(x)$

Four types of variable to describe computation of $M$ on $x$
Variables of $f_M(x)$

Four types of variable to describe computation of $M$ on $x$

- $T(b, h, i)$: tape cell at position $h$ holds symbol $b$ at time $i$. $1 \leq h \leq p(|x|)$, $b \in \Gamma$, $0 \leq i \leq p(|x|)$

$M$ is non-deterministic, need to specify transitions in some way.

Number transitions as $1, 2, \ldots, \ell$ where $j$th transition is $<q_j, b_j, q'_j, b'_j, d_j>$ indicating $(q'_j, b'_j, d_j) \in \delta(q_j, b_j)$, $d_j \in \{-1, 0, 1\}$.

Number of variables is $O(p(|x|)^2)$ where constant in $O()$ hides dependence on fixed machine $M$. 

Chan, Har-Peled, Hassanieh (UIUC) CS374 Spring 2019 29 / 42
Variables of $f_M(x)$

Four types of variable to describe computation of $M$ on $x$

- $T(b, h, i)$: tape cell at position $h$ holds symbol $b$ at time $i$.
  
  $1 \leq h \leq p(|x|)$, $b \in \Gamma$, $0 \leq i \leq p(|x|)$

- $H(h, i)$: read/write head is at position $h$ at time $i$.
  
  $1 \leq h \leq p(|x|)$, $0 \leq i \leq p(|x|)$
Four types of variable to describe computation of $M$ on $x$

- $T(b, h, i)$: tape cell at position $h$ holds symbol $b$ at time $i$.
  \[1 \leq h \leq p(|x|), \quad b \in \Gamma, \quad 0 \leq i \leq p(|x|)\]

- $H(h, i)$: read/write head is at position $h$ at time $i$.
  \[1 \leq h \leq p(|x|), \quad 0 \leq i \leq p(|x|)\]

- $S(q, i)$: state of $M$ is $q$ at time $i$.
  \[q \in Q, \quad 0 \leq i \leq p(|x|)\]
Variables of $f_M(x)$

Four types of variable to describe computation of $M$ on $x$

- $T(b, h, i)$: tape cell at position $h$ holds symbol $b$ at time $i$.
  \[1 \leq h \leq p(|x|), \ b \in \Gamma, \ 0 \leq i \leq p(|x|)\]

- $H(h, i)$: read/write head is at position $h$ at time $i$.
  \[1 \leq h \leq p(|x|), \ 0 \leq i \leq p(|x|)\]

- $S(q, i)$: state of $M$ is $q$ at time $i$ $q \in Q, \ 0 \leq i \leq p(|x|)$

- $I(j, i)$: instruction number $j$ is executed at time $i$

$M$ is non-deterministic, need to specify transitions in some way. Number transitions as $1, 2, \ldots, \ell$ where $j$th transition is $< q_j, b_j, q'_j, b'_j, d_j >$ indication $(q'_j, b'_j, d_j) \in \delta(q_j, b_j)$, direction $d_j \in \{-1, 0, 1\}$. 
Variables of $f_M(x)$

Four types of variable to describe computation of $M$ on $x$

- $T(b, h, i)$: tape cell at position $h$ holds symbol $b$ at time $i$.
  \[ 1 \leq h \leq p(|x|), \; b \in \Gamma, \; 0 \leq i \leq p(|x|) \]

- $H(h, i)$: read/write head is at position $h$ at time $i$.
  \[ 1 \leq h \leq p(|x|), \; 0 \leq i \leq p(|x|) \]

- $S(q, i)$ state of $M$ is $q$ at time $i$ $q \in Q$, $0 \leq i \leq p(|x|)$

- $I(j, i)$ instruction number $j$ is executed at time $i$

$M$ is non-deterministic, need to specify transitions in some way.
Number transitions as $1, 2, \ldots, \ell$ where $j$th transition is
\[ < q_j, b_j, q'_j, b'_j, d_j > \text{ indication } (q'_j, b'_j, d_j) \in \delta(q_j, b_j), \]
direction $d_j \in \{-1, 0, 1\}$.

Number of variables is $O(p(|x|)^2)$ where constant in $O()$ hides dependence on fixed machine $M$. 
Notation

Some abbreviations for ease of notation:

\( \bigwedge_{k=1}^{m} x_k \) means \((x_1) \land (x_2) \land \ldots \land (x_m)\)

\( \bigvee_{k=1}^{m} x_k \) means \((x_1) \lor (x_2) \lor \ldots \lor (x_m)\)

\( \bigoplus(x_1, x_2, \ldots, x_k) \) is a formula that means exactly one of \(x_1, x_2, \ldots, x_m\) is true. Can be converted to CNF form:

\( (x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2}) \)
Clauses of $f_M(x)$

$f_M(x)$ is the conjunction of 8 clause groups:

$$f_M(x) = \varphi_1 \land \varphi_2 \land \varphi_3 \land \varphi_4 \land \varphi_5 \land \varphi_6 \land \varphi_7 \land \varphi_8$$

where each $\varphi_i$ is a \textbf{CNF} formula. Described in subsequent slides.

\textbf{Property:} $f_M(x)$ is satisfied iff there is a truth assignment to the variables that simultaneously satisfy $\varphi_1, \ldots, \varphi_8$. 
φ₁ asserts (is true iff) the variables are set T/F indicating that $M$ starts in state $q₀$ at time $0$ with tape contents containing $x$ followed by blanks.

Let $x = a₁a₂\ldots aₙ$

$φ₁ = S(q₀, 0)$ state at time $0$ is $q₀$
\( \varphi_1 \) asserts (is true iff) the variables are set T/F indicating that \( M \) starts in state \( q_0 \) at time 0 with tape contents containing \( x \) followed by blanks.

Let \( x = a_1 a_2 \ldots a_n \)

\( \varphi_1 = S(q_0, 0) \) state at time 0 is \( q_0 \)
\[ \land \] and
\( \varphi_1 \) asserts (is true iff) the variables are set T/F indicating that \( M \) starts in state \( q_0 \) at time 0 with tape contents containing \( x \) followed by blanks.

Let \( x = a_1 a_2 \ldots a_n \)

\( \varphi_1 = S(q_0, 0) \) state at time 0 is \( q_0 \)
\( \land \) and
\( \land_{h=1}^{n} T(a_h, h, 0) \) at time 0 cells 1 to \( n \) have \( a_1 \) to \( a_n \)
\( \varphi_1 \) asserts (is true iff) the variables are set T/F indicating that \( M \) starts in state \( q_0 \) at time 0 with tape contents containing \( x \) followed by blanks.

Let \( x = a_1a_2 \ldots a_n \)

\( \varphi_1 = S(q_0, 0) \) state at time 0 is \( q_0 \)

\( \land \) and

\( \land_{h=1}^{n} T(a_h, h, 0) \) at time 0 cells 1 to \( n \) have \( a_1 \) to \( a_n \)

\( \land_{h=n+1}^{p(|x|)} T(B, h, 0) \) at time 0 cells \( n + 1 \) to \( p(|x|) \) have blanks
\( \varphi_1 \) asserts (is true iff) the variables are set T/F indicating that \( M \) starts in state \( q_0 \) at time 0 with tape contents containing \( x \) followed by blanks.

Let \( x = a_1 a_2 \ldots a_n \)

\( \varphi_1 = S(q_0, 0) \) state at time 0 is \( q_0 \)

\( \bigwedge \) and

\( \bigwedge_{h=1}^{n} T(a_h, h, 0) \) at time 0 cells 1 to \( n \) have \( a_1 \) to \( a_n \)

\( \bigwedge_{h=n+1}^{p(|x|)} T(B, h, 0) \) at time 0 cells \( n + 1 \) to \( p(|x|) \) have blanks

\( \bigwedge \) and

\( H(1, 0) \) head at time 0 is in position 1
\[ \varphi_2 \text{ asserts } M \text{ in exactly one state at any time } i \]

\[ \varphi_2 = \bigwedge_{i=0}^{p(|x|)} (\bigoplus (S(q_0, i), S(q_1, i), \ldots, S(q_{|Q|}, i))) \]
\( \varphi_3 \) asserts that each tape cell holds a unique symbol at any given time.

\[
\varphi_3 = \bigwedge_{i=0}^{p(|x|)} \bigwedge_{h=1}^{p(|x|)} \bigoplus (T(\bar{b}_1, h, i), T(\bar{b}_2, h, i), \ldots, T(\bar{b}_{|\Gamma|}, h, i))
\]

For each time \( i \) and for each cell position \( h \) exactly one symbol \( b \in \Gamma \) at cell position \( h \) at time \( i \)
$\varphi_4$ asserts that the read/write head of $M$ is in exactly one position at any time $i$

$$\varphi_4 = \bigwedge_{i=0}^{p(|x|)} \bigoplus (H(1, i), H(2, i), \ldots, H(p(|x|), i))$$
\( \varphi_5 \) asserts that \( M \) accepts

- Let \( q_a \) be unique accept state of \( M \)
- without loss of generality assume \( M \) runs all \( p(|x|) \) steps

\[ \varphi_5 = S(q_a, p(|x|)) \]

State at time \( p(|x|) \) is \( q_a \) the accept state.
\(\varphi_5\) asserts that \(M\) accepts

- Let \(q_a\) be unique accept state of \(M\)
- Without loss of generality assume \(M\) runs all \(p(|x|)\) steps

\[
\varphi_5 = S(q_a, p(|x|))
\]

State at time \(p(|x|)\) is \(q_a\) the accept state.

If we don’t want to make assumption of running for all steps

\[
\varphi_5 = \bigvee_{i=1}^{p(|x|)} S(q_a, i)
\]

which means \(M\) enters accepts state at some time.
$\varphi_6$ asserts that $M$ executes a unique instruction at each time

$$\varphi_6 = \bigwedge_{i=0}^{p(|x|)} \bigoplus (l(1, i), l(2, i), \ldots, l(m, i))$$

where $m$ is max instruction number.
\( \varphi_7 \) ensures that variables don’t allow tape to change from one moment to next if the read/write head was not there.

“If head is not at position \( h \) at time \( i \) then at time \( i + 1 \) the symbol at cell \( h \) must be unchanged”
\( \varphi_7 \) ensures that variables don’t allow tape to change from one moment to next if the read/write head was not there.

“If head is not at position \( h \) at time \( i \) then at time \( i + 1 \) the symbol at cell \( h \) must be unchanged”

\[
\varphi_7 = \bigwedge_{i} \bigwedge_{h} \bigwedge_{b \neq c} \left( H(h, i) \Rightarrow T(b, h, i) \bigwedge T(c, h, i + 1) \right)
\]
\( \varphi_7 \) ensures that variables don’t allow tape to change from one moment to next if the read/write head was not there.

“If head is not at position \( h \) at time \( i \) then at time \( i + 1 \) the symbol at cell \( h \) must be unchanged”

\[
\varphi_7 = \bigwedge_{i} \bigwedge_{h} \bigwedge_{b \neq c} \left( H(h, i) \implies T(b, h, i) \land T(c, h, i + 1) \right)
\]

since \( A \implies B \) is same as \( \neg A \lor B \), rewrite above in CNF form

\[
\varphi_7 = \bigwedge_{i} \bigwedge_{h} \bigwedge_{b \neq c} \left( H(h, i) \lor \neg T(b, h, i) \lor \neg T(c, h, i + 1) \right)
\]
\( \varphi_8 \) asserts that changes in tableau/tape correspond to transitions of \( M \) (as Lenny says, this is the big cookie).

Let the \( j \)th instruction be \(< q_j, b_j, q'_j, b'_j, d_j >\)
φ₈ asserts that changes in tableau/tape correspond to transitions of M (as Lenny says, this is the big cookie).

Let jth instruction be \( < q_j, b_j, q'_j, b'_j, d_j > \)

\[ φ₈ = \wedge_i \wedge_j (I(j, i) \Rightarrow S(q_j, i)) \]

If instr \( j \) executed at time \( i \) then state must be correct to do \( j \)
\( \varphi_8 \) asserts that changes in tableau/tape correspond to transitions of \( M \) (as Lenny says, this is the big cookie).

Let \( j \)th instruction be \( \langle q_j, b_j, q'_j, b'_j, d_j \rangle \)

\[ \varphi_8 = \bigwedge_i \bigwedge_j (I(j, i) \Rightarrow S(q_j, i)) \]

If instr \( j \) executed at time \( i \) then state must be correct to do \( j \).

\[ \bigwedge_i \bigwedge_j (I(j, i) \Rightarrow S(q'_j, i + 1)) \]

and at next time unit, state must be the proper next state for instr \( j \).

\[ \bigwedge_i \bigwedge_j (I(j, i) \Rightarrow T(b'_j, h, i)) \]

if \( j \) was executed and head was at position \( h \), then cell \( h \) has correct symbol for \( j \).

\[ \bigwedge_i \bigwedge_j (I(j, i) \wedge H(h, i) \Rightarrow H(h + d_j, i + 1)) \]

and head is moved properly according to instr \( j \).
\( \varphi_8 \) asserts that changes in tableau/tape correspond to transitions of \( M \) (as Lenny says, this is the big cookie).

Let \( j \)th instruction be \(< q_j, b_j, q'_j, b'_j, d_j >\)

\( \varphi_8 = \bigwedge_i \bigwedge_j (I(j, i) \Rightarrow S(q_j, i)) \) If instr \( j \) executed at time \( i \) then state must be correct to do \( j \)

\( \bigwedge_i \bigwedge_j (I(j, i) \Rightarrow S(q'_j, i + 1)) \) and at next time unit, state must be the proper next state for instr \( j \)

\( \bigwedge_i \bigwedge_h \bigwedge_j [(I(j, i) \land H(h, i)) \Rightarrow T(b_j, h, i)] \) if \( j \) was executed and head was at

position \( h \), then cell \( h \) has correct symbol for \( j \)
\( \phi_8 \) asserts that changes in tableau/tape correspond to transitions of \( M \) (as Lenny says, this is the big cookie).

Let \( j \)th instruction be \(< q_j, b_j, q'_j, b'_j, d_j >\)

\[
\phi_8 = \bigwedge_i \bigwedge_j (I(j, i) \Rightarrow S(q_j, i)) \quad \text{If instr } j \text{ executed at time } i \text{ then state must be correct to do } j
\]
\[
\bigwedge_i \bigwedge_j (I(j, i) \Rightarrow S(q'_j, i + 1)) \quad \text{and at next time unit, state must be the proper next state for instr } j
\]
\[
\bigwedge_i \bigwedge_h \bigwedge_j [(I(j, i) \wedge H(h, i)) \Rightarrow T(b_j, h, i)] \quad \text{if } j \text{ was executed and head was at position } h, \text{ then cell } h \text{ has correct symbol for } j
\]
\[
\bigwedge_i \bigwedge_j \bigwedge_h [(I(j, i) \wedge H(h, i)) \Rightarrow T(b'_j, h, i + 1)] \quad \text{if } j \text{ was done then at time } i \text{ with head at } h \text{ then at next time step symbol } b'_j \text{ was indeed written in position } h
\( \varphi_8 \) asserts that changes in tableau/tape correspond to transitions of \( M \) (as Lenny says, this is the big cookie).

Let the \( j \)th instruction be \(< q_j, b_j, q'_j, b'_j, d_j >\)

\[
\varphi_8 = \bigwedge_i \bigwedge_j (I(j, i) \Rightarrow S(q_j, i)) \quad \text{If instr } j \text{ executed at time } i \text{ then state must be correct to do } j
\]
\[
\bigwedge_i \bigwedge_j (I(j, i) \Rightarrow S(q'_j, i + 1)) \quad \text{and at next time unit, state must be the proper next state for instr } j
\]
\[
\bigwedge_i \bigwedge_h \bigwedge_j [(I(j, i) \wedge H(h, i)) \Rightarrow T(b_j, h, i)] \quad \text{if } j \text{ was executed and head was at position } h, \text{ then cell } h \text{ has correct symbol for } j
\]
\[
\bigwedge_i \bigwedge_j \bigwedge_h [(I(j, i) \wedge H(h, i)) \Rightarrow T(b'_j, h, i + 1)] \quad \text{if } j \text{ was done then at time } i \text{ with head at } h \text{ then at next time step symbol } b'_j \text{ was indeed written in position } h
\]
\[
\bigwedge_i \bigwedge_j \bigwedge_h [(I(j, i) \wedge H(h, i)) \Rightarrow H(h + d_j, i + 1)] \quad \text{and head is moved properly according to instr } j.\]
Clauses of $f_M(x)$

$f_M(x)$ is the conjunction of 8 clause groups:

$$f_M(x) = \varphi_1 \land \varphi_2 \land \varphi_3 \land \varphi_4 \land \varphi_5 \land \varphi_6 \land \varphi_7 \land \varphi_8$$

where each $\varphi_i$ is a CNF formula.
Clauses of $f_M(x)$

$f_M(x)$ is the conjunction of 8 clause groups:

$$f_M(x) = \varphi_1 \land \varphi_2 \land \varphi_3 \land \varphi_4 \land \varphi_5 \land \varphi_6 \land \varphi_7 \land \varphi_8$$

where each $\varphi_i$ is a CNF formula.

$\varphi_1$ asserts $M$ starts in state $q_0$ at time 0 with tape contents containing $x$ followed by blanks.
Clauses of $f_M(x)$

$f_M(x)$ is the conjunction of 8 clause groups:

$$f_M(x) = \varphi_1 \land \varphi_2 \land \varphi_3 \land \varphi_4 \land \varphi_5 \land \varphi_6 \land \varphi_7 \land \varphi_8$$

where each $\varphi_i$ is a **CNF** formula.

$\varphi_1$ asserts $M$ starts in state $q_0$ at time 0 with tape contents containing $x$ followed by blanks.

$\varphi_2$ asserts $M$ in exactly one state at any time.
Clauses of $f_M(x)$

$f_M(x)$ is the conjunction of 8 clause groups:

$$f_M(x) = \varphi_1 \land \varphi_2 \land \varphi_3 \land \varphi_4 \land \varphi_5 \land \varphi_6 \land \varphi_7 \land \varphi_8$$

where each $\varphi_i$ is a CNF formula.

$\varphi_1$ asserts $M$ starts in state $q_0$ at time 0 with tape contents containing $x$ followed by blanks.

$\varphi_2$ asserts $M$ in exactly one state at any time.

$\varphi_3$ asserts that each tape cell holds a unique symbol at any time.
Clauses of $f_M(x)$

$f_M(x)$ is the conjunction of 8 clause groups:

$$f_M(x) = \varphi_1 \land \varphi_2 \land \varphi_3 \land \varphi_4 \land \varphi_5 \land \varphi_6 \land \varphi_7 \land \varphi_8$$

where each $\varphi_i$ is a CNF formula.

$\varphi_1$ asserts $M$ starts in state $q_0$ at time 0 with tape contents containing $x$ followed by blanks.

$\varphi_2$ asserts $M$ in exactly one state at any time.

$\varphi_3$ asserts that each tape cell holds a unique symbol at any time.

$\varphi_4$ asserts that the head of $M$ is in exactly one position at any time.
Clauses of \(f_M(x)\)

\(f_M(x)\) is the conjunction of 8 clause groups:

\[
f_M(x) = \varphi_1 \land \varphi_2 \land \varphi_3 \land \varphi_4 \land \varphi_5 \land \varphi_6 \land \varphi_7 \land \varphi_8
\]

where each \(\varphi_i\) is a CNF formula.

\(\varphi_1\) asserts \(M\) starts in state \(q_0\) at time 0 with tape contents containing \(x\) followed by blanks.

\(\varphi_2\) asserts \(M\) in exactly one state at any time.

\(\varphi_3\) asserts that each tape cell holds a unique symbol at any time.

\(\varphi_4\) asserts that the head of \(M\) is in exactly one position at any time.

\(\varphi_5\) asserts that \(M\) accepts.
Clauses of $f_M(x)$

$f_M(x)$ is the conjunction of 8 clause groups:

$$f_M(x) = \varphi_1 \land \varphi_2 \land \varphi_3 \land \varphi_4 \land \varphi_5 \land \varphi_6 \land \varphi_7 \land \varphi_8$$

where each $\varphi_i$ is a CNF formula.

$\varphi_1$ asserts $M$ starts in state $q_0$ at time 0 with tape contents containing $x$ followed by blanks.

$\varphi_2$ asserts $M$ in exactly one state at any time.

$\varphi_3$ asserts that each tape cell holds a unique symbol at any time.

$\varphi_4$ asserts that the head of $M$ is in exactly one position at any time.

$\varphi_5$ asserts that $M$ accepts.

$\varphi_6$ asserts that $M$ executes a unique instruction at each time.
Clauses of $f_M(x)$

$f_M(x)$ is the conjunction of 8 clause groups:

$$f_M(x) = \varphi_1 \land \varphi_2 \land \varphi_3 \land \varphi_4 \land \varphi_5 \land \varphi_6 \land \varphi_7 \land \varphi_8$$

where each $\varphi_i$ is a CNF formula.

$\varphi_1$ asserts $M$ starts in state $q_0$ at time 0 with tape contents containing $x$ followed by blanks.

$\varphi_2$ asserts $M$ in exactly one state at any time.

$\varphi_3$ asserts that each tape cell holds a unique symbol at any time.

$\varphi_4$ asserts that the head of $M$ is in exactly one position at any time.

$\varphi_5$ asserts that $M$ accepts.

$\varphi_6$ asserts that $M$ executes a unique instruction at each time.

$\varphi_7$ ensures that variables don’t allow tape to change from one moment to next if the read/write head was not there.
Clauses of $f_M(x)$

$f_M(x)$ is the conjunction of 8 clause groups:

$$f_M(x) = \varphi_1 \land \varphi_2 \land \varphi_3 \land \varphi_4 \land \varphi_5 \land \varphi_6 \land \varphi_7 \land \varphi_8$$

where each $\varphi_i$ is a CNF formula.

$\varphi_1$ asserts $M$ starts in state $q_0$ at time 0 with tape contents containing $x$ followed by blanks.

$\varphi_2$ asserts $M$ in exactly one state at any time.

$\varphi_3$ asserts that each tape cell holds a unique symbol at any time.

$\varphi_4$ asserts that the head of $M$ is in exactly one position at any time.

$\varphi_5$ asserts that $M$ accepts.

$\varphi_6$ asserts that $M$ executes a unique instruction at each time.

$\varphi_7$ ensures that variables don’t allow tape to change from one moment to next if the read/write head was not there.

$\varphi_8$ asserts that changes in tableau/tape correspond to transitions of $M$.
Proof of Correctness

(Sketch)

- Given $M$, $x$, poly-time algorithm to construct $f_M(x)$
- if $f_M(x)$ is satisfiable then the truth assignment completely specifies an accepting computation of $M$ on $x$
- if $M$ accepts $x$ then the accepting computation leads to an "obvious" truth assignment to $f_M(x)$. Simply assign the variables according to the state of $M$ and cells at each time $i$.

Thus $M$ accepts $x$ if and only if $f_M(x)$ is satisfiable
<table>
<thead>
<tr>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. SAT</td>
</tr>
<tr>
<td>2. 3SAT</td>
</tr>
<tr>
<td>3. CircuitSAT</td>
</tr>
<tr>
<td>4. Independent Set</td>
</tr>
<tr>
<td>5. Clique</td>
</tr>
<tr>
<td>6. Vertex Cover</td>
</tr>
<tr>
<td>7. <strong>Hamilton Cycle</strong> and <strong>Hamilton Path</strong> in both directed and undirected graphs</td>
</tr>
<tr>
<td>8. 3Color and Color</td>
</tr>
</tbody>
</table>