Computer Science 511

The Cook-Levin Theorem

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #11

Goals for Today

- Introduce a "Boolean Formula Satisfiability" Problem and prove that its associated language, L_{FSAT}, is $\mathcal{NP}$-complete.
- The proof in these notes is *not* (quite) the proof contained in Cook's paper. It is based on the proof given as the proof of Theorem 7.37 in the third edition of Michael Sipser's text, *Introduction to the Theory of Computation.*

Boolean Formulas: Definition

Consider a set

$$\mathcal{V} = \{x_0, x_1, x_2, \ldots\}$$

of ***Boolean variables*** — each of which can have value ***true*** (T) or ***false*** (F).

Definition: A ***Boolean formula*** over these variables is defined using the following rules.

1. x_i is a Boolean formula, for every nonnegative integer i.
2. If $\mathcal{F}$ is a Boolean formula then $\neg\mathcal{F}$ is also a Boolean formula.
3. If $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k$ are Boolean formulas, for an integer $k \geq 2$, then the following are Boolean formulas as well:
   (a) $(\mathcal{F}_1 \wedge \mathcal{F}_2 \wedge \cdots \wedge \mathcal{F}_k)$
   (b) $(\mathcal{F}_1 \vee \mathcal{F}_2 \vee \cdots \vee \mathcal{F}_k)$

Boolean Formulas: Values under Truth Assignments

A ***partial truth assignment*** is a partial function

$$\varphi : \mathcal{V} \to \{\mathtt{T}, \mathtt{F}\}$$

— so $\varphi(x_i)$ is either $\mathtt{T}$, $\mathtt{F}$, or "undefined" for every natural number i.

A ***truth assignment*** (or "***total truth assignment***") is a partial truth assignment φ such that either $\varphi(x_i) = \mathtt{T}$ or $\varphi(x_i) = \mathtt{F}$ for every natural number n. That is, $\varphi(x_i)$ is never "undefined".

Boolean Formulas: Values under Truth Assignments

Truth values for Boolean formulas under partial truth assignments can be inductively defined: Let $\varphi : \mathcal{V} \to \{\mathtt{T}, \mathtt{F}\}$ be a partial truth assignment and let $\mathcal{F}$ be a Boolean formula.

1. If $\mathcal{F}_i$ is the Boolean formula x_i, for a natural number i, then $\varphi(\mathcal{F}) = \varphi(x_i)$.
2. If $\mathcal{F} = \neg\widehat{\mathcal{F}}$ for a Boolean formula $\neg\widehat{\mathcal{F}}$ then $\varphi(\mathcal{F})$ is undefined if $\varphi(\widehat{\mathcal{F}})$ is undefined, and $\varphi(\mathcal{F}) = \neg\varphi(\widehat{\mathcal{F}})$ otherwise.

Boolean Formulas: Values under Truth Assignments

3. Let k be an integer such that $k \geq 2$ and let $\widehat{\mathcal{F}}_1, \widehat{\mathcal{F}}_2, \ldots, \widehat{\mathcal{F}}_k$ be Boolean formulas.
   (a) Suppose that $\mathcal{F} = (\widehat{\mathcal{F}}_1 \wedge \widehat{\mathcal{F}}_2 \wedge \cdots \wedge \widehat{\mathcal{F}}_k)$. If $\varphi(\widehat{\mathcal{F}}_i)$ is undefined, for any integer i such that $1 \leq i \leq k$, then $\varphi(\mathcal{F})$ is undefined as well. Otherwise

$$\varphi(\mathcal{F}) = (\varphi(\widehat{\mathcal{F}}_1) \wedge \varphi(\widehat{\mathcal{F}}_2) \wedge \cdots \wedge \varphi(\widehat{\mathcal{F}}_k)).$$

   (b) Suppose that $\mathcal{F} = (\widehat{\mathcal{F}}_1 \vee \widehat{\mathcal{F}}_2 \vee \cdots \vee \widehat{\mathcal{F}}_k)$. If $\varphi(\widehat{\mathcal{F}}_i)$ is undefined, for any integer i such that $1 \leq i \leq k$, then $\varphi(\mathcal{F})$ is undefined as well. Otherwise

$$\varphi(\mathcal{F}) = (\varphi(\widehat{\mathcal{F}}_1) \vee \varphi(\widehat{\mathcal{F}}_2) \vee \cdots \vee \varphi(\widehat{\mathcal{F}}_k)).$$

Boolean Formulas: Values under Truth Assignments

Definition: A Boolean formula $\mathcal{F}$ is ***satisfiable*** if there exists a truth assignment[1] $\varphi : \mathcal{V} \to \{\mathtt{T}, \mathtt{F}\}$ such that $\varphi(\mathcal{F}) = \mathtt{T}$.

We say that $\mathcal{F}$ is ***satisfied*** under a truth assignment $\varphi : \mathcal{V} \to \{\mathtt{T}, \mathtt{F}\}$ if $\varphi(\mathcal{F}) = \mathtt{T}$.

Note: These definitions, concerning "satisfiability", should be the same as in any previous courses in mathematics or logic, which students have taken, in which Boolean formulas are defined — but they are possibly presented somewhat more formally, here.

[1] Partial truth assignments can be used instead of (total) truth assignments in this definition without changing the set of "satisfiable" Boolean formulas.

Boolean Formulas: Encodings

Let

$$\Sigma_F = \{\mathtt{x}, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \wedge, \vee, \neg, (,)\}.$$

An encoding $\mathrm{e}(\mathcal{F})$ as a string in $\Sigma_F^\star$ can also be inductively defined.

1. For every nonnegative integer i the encoding $\mathrm{e}(x_i)$ of the Boolean variable x_i is the string beginning with the symbol $\mathtt{x}$, followed by the unpadded decimal representation of i. Thus if $\mathcal{F} = x_i$ then $\mathrm{e}(\mathcal{F})$ is this string, $\mathrm{e}(x_i)$, as well.
2. If $\mathcal{F} = \neg\widehat{\mathcal{F}}$ for a Boolean formula $\widehat{\mathcal{F}}$ then $\mathrm{e}(\mathcal{F})$ is the symbol $\neg \in \Sigma_F$ followed by the encoding $\mathrm{e}(\widehat{\mathcal{F}})$ of $\widehat{\mathcal{F}}$.

Boolean Formulas: Encodings

3. Suppose that k is an integer such that $k \geq 2$ and that $\widehat{\mathcal{F}}_1, \widehat{\mathcal{F}}_2, \ldots, \widehat{\mathcal{F}}_k$ are Boolean formulas.
   (a) If $\mathcal{F} = (\widehat{\mathcal{F}}_1 \wedge \widehat{\mathcal{F}}_2 \wedge \cdots \wedge \widehat{\mathcal{F}}_k)$ then $e(\mathcal{F})$ is the string consisting of the encodings $e(\widehat{\mathcal{F}}_1), e(\widehat{\mathcal{F}}_2), \ldots, e(\widehat{\mathcal{F}}_k)$, separated by copies of the symbol $\wedge \in \Sigma_F$, and enclosed by brackets (that is, starting with "(" and ending with ")").
   (b) If $\mathcal{F} = (\widehat{\mathcal{F}}_1 \vee \widehat{\mathcal{F}}_2 \vee \cdots \vee \widehat{\mathcal{F}}_k)$ then $e(\mathcal{F})$ is the string consisting of the encodings $e(\widehat{\mathcal{F}}_1), e(\widehat{\mathcal{F}}_2), \ldots, e(\widehat{\mathcal{F}}_k)$, separated by copies of the symbol $\vee \in \Sigma_F$, and enclosed by brackets (that is, starting with "(" and ending with ")").

Boolean Formulas: Encodings

In other words, $e(\mathcal{F})$ is produced from $\mathcal{F}$ by replacing each Boolean variable x_i with the string $e(x_i) \in \Sigma_F^\star$ that encodes it, and encoding punctuation ("(" and ")") and operators ("$\neg$", "$\vee$" and "$\neg$") by the corresponding symbols in Σ_F.

Boolean Formulas: Languages of Interest

- Let $L_F \subseteq \Sigma_F^\star$ be the set of encodings of Boolean formulas, as defined above.
- Let L_{FSAT} be the set of encodings of ***satisfiable*** Boolean formulas, as defined above.

$L_F \in \mathcal{P}$

Claim #1: $L_F \in \mathcal{P}$.

How To Prove This: Consider a deterministic Turing machine that begins by sweeping over its input string (while copying it onto another tape), replacing each encoding of a Boolean variable that is seen by a single symbol, "$\mathtt{F}$".

Repeatedly replacing substrings with one of the forms

- $\neg\mathtt{F}$
- $(\mathtt{F} \wedge \mathtt{F} \wedge \cdots \wedge \mathtt{F})$
- $(\mathtt{F} \vee \mathtt{F} \vee \cdots \vee \mathtt{F})$

with $\mathtt{F}$, one eventually obtains a string with length one — $\mathtt{F}$ — if $\omega \in L_F$, and the input can be ***accepted***. If $\omega \notin L_F$ then this process fails (in one of several predictable ways) and the input can be ***rejected*** instead.

$L_F \in \mathcal{P}$

- While it might seem plausible that a Turing machine, that implements this, would decide the language L_F, it might not be clear how to prove this.
- The supplement for this lecture includes more information about how one can prove that this Turing machine *would* decide the language L_F, using a number of steps that is at most quadratic in the length of the input string in the worst case — implying that $L_F \in \mathcal{P}$.

$L_{FSAT} \in \mathcal{NP}$

Claim #2: $L_{FSAT} \in \mathcal{NP}$.

How To Prove This: A polynomial-time verification algorithm for this language should be described and analyzed.

- If $\omega \in \Sigma_F^\star$ is an encoding of a satisfiable Boolean formula $\mathcal{F}$ (so that $\omega \in L_F$) then a ***certificate*** for ω will be an encoding of a truth assignment $\varphi : \mathcal{V} \to \{\mathtt{T}, \mathtt{F}\}$ that satisfies $\mathcal{F}$ — that is, such that $\varphi(\mathcal{F}) = \mathtt{T}$.
- Truth assignments will be encoded using a ***certificate alphabet***

$$\Sigma_C = \{\mathtt{x}, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \{, \}, ,\}$$

including the symbols in Σ_F used to encode Boolean variables, set brackets ("$\{$" and "$\}$"), and a comma.

$L_{FSAT} \in \mathcal{NP}$

- Suppose $\varphi : \mathcal{V} \to \{\mathtt{T}, \mathtt{F}\}$ is a truth assignment. Then — for the purposes of this problem, φ will be encoded by a string $\mathrm{e}(\varphi) \in \Sigma_C^\star$, including the encoding $\varphi(x_i)$ of every variable $x_i \in \mathcal{V}$ such that
  - x_i appears in $\mathcal{F}$ (so that its truth value is needed), and
  - $\varphi(x_i) = \mathtt{T}$.

  These encodings of variables should be separated by commas, and enclosed by set brackets (so that $\mathrm{e}(\varphi)$ begins with "$\{$" and ends with "$\}$"). The encodings of variables x_i, included here, should be sorted by increasing index i.

$L_{FSAT} \in \mathcal{NP}$

With a bit of work, each of the following can be proved.

- If $\omega \in \Sigma_F^\star$ is an coding of a satisfiable Boolean formula then there exists a string $\mu \in \Sigma_C^\star$ that is a certificate for ω, as described above, such that $|\mu| \leq |\omega| + 2$.
- Given a string $\omega \in L_F$ — so that ω encodes a Boolean formula $\mathcal{F}$ — and given a string $\nu \in \Sigma_C^\star$ such that $|\nu| \leq |\omega| + 2$, it is possible to decide whether ν encodes a set $\mathcal{S}$ of Boolean variables such that
  - each Boolean variable x_i in $\mathcal{S}$ appears in $\mathcal{F}$, and
  - the encodings of variables, in $\mathcal{S}$, are sorted by increasing index,

  using a deterministic Turing machine, using a number of moves that is at most polynomial in $|\omega|$.

Proving these claims is left as an ***exercise***.

$L_{FSAT} \in \mathcal{NP}$

Consider a deterministic Turing machine with input alphabet $\widehat{\Sigma} = \Sigma_F \cup \Sigma_C \cup \{\#\}$ that implements an algorithm with the following major steps.

1. ***Reject*** if the input string does not begin with a string with the form $\omega\#$, where $\omega \in L_F$, copying ω onto the second tape otherwise. Let $\mathcal{F}$ be the Boolean formula encoded by $\mathcal{F}$ if the input has not been rejected.
2. ***Reject*** if the rest of the input string is *not* a string in $\Sigma_C^\star$ with length at most $|\omega| + 2$ that encodes a truth assignment $\varphi : \mathcal{V} \to \{\mathtt{T}, \mathtt{F}\}$.
3. Compute the truth value of $\mathcal{F}$ under φ. ***Accept*** if $\varphi(\mathcal{F}) = \mathtt{T}$, and ***reject*** otherwise.

$L_{FSAT} \in \mathcal{NP}$

- With a bit of work this can be shown to be a verification algorithm for L_{FSAT} that can be implemented using a deterministic Turing machine M, such that the number of moves used by M is at most a polynomial function of $|\omega|$ whenever the input has the form $\omega\#\nu$ where $\omega \in \Sigma_F^\star$ and $\nu \in \Sigma_C^\star$: See the supplemental material for details.
- It follows that $L_{FSAT} \in \mathcal{NP}$, as claimed.

L_{FSAT} is $\mathcal{NP}$-Hard: Overview of Proof

Claim #3: L_{FSAT} is $\mathcal{NP}$-hard.

How To Prove This: Prove that $L \preceq_{P,\,M} L_{FSAT}$ for an *arbitrarily chosen* language $L \in \mathcal{NP}$.

- Let $L \subseteq \Sigma^\star$ such that $L \in \mathcal{NP}$.
- Then there exists a ***nondeterministic one-tape*** Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

deciding L, and there exist positive integer constants d, c_1 and c_0 such that depth of the computation tree for M and ω is at most

$$T(|\omega|) = c_1 |\omega|^d + c_0,$$

for every string $\omega \in \Sigma^\star$.

L_{FSAT} is $\mathcal{NP}$-Hard: Overview of Proof

- A few ***simplifying assumptions***[2] can be made to make the proof easier:
  - Adding transitions to the rejecting state when needed (and adding one to c_0), it can be assumed that $\delta(q, \sigma)$ is not the empty set whenever $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ and $\sigma \in \Gamma$.
  - Renaming states as needed, it can be assumed that $Q \cap \Gamma = \emptyset$.
  - Increasing c_0 as needed, it can be assumed that $c_0 \geq 4$.

[2] These can, generally, be ignored when reading these notes. They will be helpful for the proofs in the supplemental material.

L_{FSAT} is $\mathcal{NP}$-Hard: Overview of Proof

- As described for deterministic Turing machines, a ***configuration*** of the nondeterministic Turing machine M is a string

$$\mu_1 q \mu_2$$

  where $\mu_1, \mu_2 \in \Gamma^\star$ and $q \in Q$.
- This represents the fact that the non-blank part of the tape currently stores the string $\mu_1\mu_2$ — beginning with the symbol stored at the leftmost cell — and that the tape head currently points to the leftmost symbol in μ_2 (or the leftmost blank to the right of μ_1, if $\mu_2 = \lambda$).

L_{FSAT} is $\mathcal{NP}$-Hard: Overview of Proof

- If at most t moves have been made during the execution of M on input ω, then M is in configuration

$$\mu_1 q \mu_2,$$

then $|\mu_1| \leq t$ and — after removing trailing blanks — one could require that $|\mu_2| \leq \max(t, |\omega|)$.

- For the construction in this proof, we will ***pad*** configurations by adding blanks, to the right of μ_2, in order to ensure that

$$|\mu_1| + |\mu_2| = T(|\omega|) + 1 = c_1 |\omega|^d + c_0 + 1$$

for every configuration $\mu_1 q \mu_2$ being considered.

L_{FSAT} is $\mathcal{NP}$-Hard: Overview of Proof

Definition: Let $\omega \in \Sigma^\star$ and let $n = |\omega|$. A ***tableau*** for ***M*** on input ω is a two-dimensional table with $T(n) + 1$ rows and $T(n) + 4$ columns that satisfies the following properties.

(a) Each cell of the table stores a single symbol in $Q \cup \Gamma \cup \{\#\}$.

(b) When read in order from left to right, the symbols in each row form a string

$$\#\mu_1 q \mu_2 \#$$

where $\mu_1, \mu_2 \in \Gamma^\star$ and $q \in Q$ — so that $\mu_1 q \mu_2$ is a (possible) configuration of M. The tableau has enough columns for any configuration that *might* arise, during the execution of M on input ω, to be represented, in a row of the tableau, in this way.

L_{FSAT} is $\mathcal{NP}$-Hard: Overview of Proof

(c) If

$$\omega = \sigma_1 \sigma_2 \ldots \sigma_n$$

(where $\sigma_1, \sigma_2, \ldots, \sigma_n \in \Sigma$) then the cells in the top row (row #0) of the tableau store the string

$$\#q_0 \sigma_1 \sigma_2 \ldots \sigma_n \sqcup \sqcup \ldots \sqcup \#$$

— the (padded) representation for the ***initial*** configuration of M and the input string ω.

L_{FSAT} is $\mathcal{NP}$-Hard: Overview of Proof

(d) For $0 \leq i \leq T(n) - 1$, if η_i is the configuration represented in row i and η_{i+1} is the configuration represented in row $i + 1$, then either

- $\eta_i \vdash_M \eta_{i+1}$, that is, it is possible to go from configuration η_i to configuration η_{i+1} using a single move of M,

or

- it is not possible for M to move out of configuration η_i — because this is either an accepting configuration or a rejecting configuration — and $\eta_i = \eta_{i+1}$.

Thus the tableau gives a ***trace of one possible execution*** of M on the input string ω — including enough rows for an *entire* trace of execution (ending with either an accepting configuration or a rejecting configuration in the bottom row) to be shown.

L_{FSAT} is $\mathcal{NP}$-Hard: Overview of Proof

- A tableau is an ***accepting tableau*** if the configuration represented by the bottom row is an accepting configuration — that is, it is in state q_{accept}.
- ***Note:*** It follows that $\omega \in L$ if and only if there exists an ***accepting configuration*** for M and ω.

L_{FSAT} is $\mathcal{NP}$-Hard: Overview of Proof

The following *could* be done to complete the proof.

- A mapping from strings $\omega \in \Sigma^\star$ to Boolean formulas $\mathcal{F}_\omega$ is described — and it is argued that, for all $\omega \in \Sigma^\star$, ω is in the language L of M if and only if the corresponding Boolean formula $\mathcal{F}_\omega$ is satisfiable.
- Reviewing information about the *encodings* of Boolean formulas, as needed, this is used to define a total function $f : \Sigma^\star \to \Sigma_F^\star$ such that, for all $\omega \in \Sigma^\star$, $\omega \in L$ if and only if $f(\omega) \in L_{FSAT}$ — because $f(\omega)$ is the encoding of the Boolean formula, $\mathcal{F}_\omega$, described above.

L_{FSAT} is $\mathcal{NP}$-Hard: Overview of Proof

- Reviewing the *definition* of the Boolean formula $\mathcal{F}_\omega$ and the above function f as needed, it is argued that there exists a ***deterministic*** Turing machine M_f that computes the function f, using a number of moves that is at most a polynomial function of the length of its input.
- Thus f is a polynomial-time many-one reduction from L to L_{FSAT}, so that $L \preceq_{P,\,M} L_{FSAT}$.
- Since L was arbitrarily chosen from $\mathcal{NP}$, it follows that L_{FSAT} is $\mathcal{NP}$-hard, as claimed.

L_{FSAT} is $\mathcal{NP}$-Hard: Overview of Proof

- ***Problem:*** There is an awful lot to do here — and it is easy to "get lost in the proof" because it is so long!
- The lecture notes include selected *pieces* of the rest of the proof, with most of the rest included in the supplemental material for this lecture.
- Ideally, students will believe that the result is true and have some understanding of the *organization* of the proof without, necessarily, having seen (or understood) all the details.
- Of course, students *will not* be asked to give details about this proof if they have not been included in these lecture notes.

L_{FSAT} is $\mathcal{NP}$-Hard: Construction of $\mathcal{F}_\omega$ from ω

In order to ***improve readability*** $\mathcal{F}_\omega$ will be described as if it was a function of Boolean variables $x_{i,j,\sigma}$ where

- i is an index of a ***row*** in a tableau for ω, so $i \in \mathbb{N}$ and $0 \leq i \leq T(n)$.
- j is an index of a ***column*** in a tableau for ω, so that $j \in \mathbb{N}$ and $0 \leq j \leq T(n) + 3$.
- σ is a symbol that might be stored in the cell in row i and column j of the tableau, so

$$\sigma \in Q \cup \Gamma \cup \{\#\}.$$

- It might help of thinking that $x_{i,j,\sigma}$ is ***true*** (that is, has value $\mathtt{T}$ under a truth assignment) if the symbol σ is stored in row i and column j of the tableau.

L_{FSAT} is $\mathcal{NP}$-Hard: Construction of $\mathcal{F}_\omega$ from ω

- Suppose we order the elements[3] of $Q \cup \Gamma \cup \{\#\}$, so that we can write

$$Q \cup \Gamma \cup \{\#\} = \{\sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_{\ell-1}\}$$

where $\ell = |Q| + |\Gamma| + 1$.

[3] listing states first, followed by symbols in Γ, and ending with #, in order to simplify details later

L_{FSAT} is $\mathcal{NP}$-Hard: Construction of $\mathcal{F}_\omega$ from ω

- For $0 \leq i \leq T(n)$, $0 \leq j \leq T(n)+3$, and $0 \leq h \leq \ell - 1$, let

$$g(\mathtt{x}_{i,j,\sigma_h}) = \mathtt{x}_{h+j\cdot\ell+i\cdot\ell\cdot(T(n)+4)\cdot}$$

  This provides an easily computable ***bijection*** from the set of variables $\mathtt{x}_{i,j,\sigma}$ (being used, here, for readability) to the set of variables $\{x_a \mid 0 \leq a < \ell \cdot (T(n)+4) \cdot (T(n)+1)\}$.
- We can apply this "on the fly" to obtain the kind of Boolean formula defined near the beginning of the lecture.

L_{FSAT} is $\mathcal{NP}$-Hard: Construction of $\mathcal{F}_\omega$ from ω

$\mathcal{F}_\omega$ is a Boolean formula with the form

$$(\mathcal{F}_{\text{cell}} \wedge \mathcal{F}_{\text{start}} \wedge \mathcal{F}_{\text{accept}} \wedge \mathcal{F}_{\text{move}}) \tag{1}$$

- $\mathcal{F}_{\text{cell}}$ represents the requirement that there is ***exactly*** one symbol from $Q \cup \Gamma \cup \{\#\}$ in each cell of the tableau.
- $\mathcal{F}_{\text{start}}$ represents the requirement that the top row of the tableau represents the initial configuration for M and the input string ω.
- $\mathcal{F}_{\text{accept}}$ represents the requirement that the *bottom* row of the tableau represents an accepting configuration.

L_{FSAT} is $\mathcal{NP}$-Hard: Construction of $\mathcal{F}_\omega$ from ω

- $\mathcal{F}_{\text{move}}$ — the most complicated of these subformulas — represents the requirement that *every* row of the tableau represents a configuration and, furthermore, if $i \in \mathbb{N}$ and $0 \leq i \leq T(n)$, then either
  - Rows i and $i+1$ represents configurations η_i and η_{i+1}, respectively, such that $\eta_i \vdash_M \eta_{i+1}$, or
  - Row i represents a ***halting*** configuration η_i, and row $i+1$ represents η_i too.

Consequence: If $f : \Sigma^\star \to \Sigma_F^\star$ is the function mapping an input string $\omega \in \Sigma^\star$ to the encoding of the corresponding Boolean formula $\mathcal{F}_\omega$, then $\omega \in L$ if and only if $f(\omega) \in L_{FSAT}$, for all $\omega \in \Sigma^\star$, as desired.

L_{FSAT} is $\mathcal{NP}$-Hard: Construction of $\mathcal{F}_\omega$ from ω

- It follows by the description of $\mathcal{F}_\omega$ at line (1) that

$$\mathsf{e}(\mathcal{F}_\omega) = (\mathsf{e}(\mathcal{F}_{\mathsf{cell}}) \wedge \mathsf{e}(\mathcal{F}_{\mathsf{start}}) \wedge \mathsf{e}(\mathcal{F}_{\mathsf{accept}}) \wedge \mathsf{e}(\mathcal{F}_{\mathsf{move}})) \in \Sigma_F^\star.$$

- If each of $\mathsf{e}(\mathcal{F}_{\mathsf{cell}})$, $\mathsf{e}(\mathcal{F}_{\mathsf{start}})$, $\mathsf{e}(\mathcal{F}_{\mathsf{accept}})$ and $\mathsf{e}(\mathcal{F}_{\mathsf{move}})$ can be computed deterministically from ω, using time polynomial in the length of ω, then $\mathsf{e}(\mathcal{F}_\omega)$ can be computed deterministically from ω, in polynomial time, too — as needed to complete the proof.

L_{FSAT} is $\mathcal{NP}$-Hard: The Subformula $\mathcal{F}_{cell}$

$\mathcal{F}_{cell}$ represents the requirement that there is ***exactly*** one symbol from $Q \cup \Gamma \cup \{\#\}$ in each cell of the tableau.

- Let $\mathcal{F}_{cell}$ be the formula

$$\bigwedge_{0 \le i \le T(n)} \left(\bigwedge_{0 \le j \le T(n)+3} \left(\left(\bigvee_{\sigma \in Q \cup \Gamma \cup \{\#\}} x_{i,j,\sigma} \right) \wedge \left(\bigwedge_{\substack{\sigma_1, \sigma_2 \in Q \cup \Gamma \cup \{\#\} \\ \sigma_1 \neq \sigma_2}} \left(\neg x_{i,j,\sigma_1} \vee \neg x_{i,j,\sigma_2} \right) \right) \right) \right)$$

It follows, by inspection of the structure of this subformula, that it is satisfied (under a truth assignment) if and only if the above requirement is met.

L_{FSAT} is $\mathcal{NP}$-Hard: The Subformula $\mathcal{F}_{cell}$

- Since M is a ***fixed*** nondeterministic Turing machine (where this construction is concerned) and the function T is a ***fixed*** polynomial function of its input n, one can argue both that the length of $e(\mathcal{F}_{cell})$ is bounded by a polynomial function of n and that the string $e(\mathcal{F}_{cell})$ (which depends only on $|\omega|$) can be computed deterministically from ω, using time bounded by a polynomial function of n as well.
- ***Suggested Exercise:*** If this is not clear, write a Java program that generates $e(\mathcal{F}_{cell})$ from $n = |\omega|$, using ℓ and the integers d, c_0 and c_1 used to define the function f as constants.

L_{FSAT} is $\mathcal{NP}$-Hard: The Subformula $\mathcal{F}_{start}$

$\mathcal{F}_{start}$ represents the requirement that the top row of the tableau represents the initial configuration for M and the input string ω.

- Renaming symbols, suppose that $\omega \in \tau_1\tau_2\ldots,\tau_n$ for $\tau_1, \tau_2, \ldots, \tau_n \in \Sigma$; then let $\mathcal{F}_{start}$ be the formula

$$x_{0,0,\#} \wedge x_{0,1,q_0} \wedge \left(\bigwedge_{1 \leq h \leq n} x_{0,h+1,\tau_h} \right) \wedge \left(\bigwedge_{n+2 \leq h \leq T(n)+2} x_{0,h,\sqcup} \right) \wedge x_{0,T(n)+3,\#}$$

Once again, it follows by inspection of this formula that it is satisfied (under a truth assignment) if and only if the above requirement is met.

L_{FSAT} is $\mathcal{NP}$-Hard: The Subformula $\mathcal{F}_{start}$

- As in the previous case, it is not difficult to argue that the length of the string $e(\mathcal{F}_{start})$ is bounded by a polynomial function of $|\omega|$ and — because the structure of this subformula is reasonably simple and uniform — the string $e(\mathcal{F}_{start})$ can be computed from ω deterministically, using a number of moves bounded by a polynomial function of $|\omega|$.

L_{FSAT} is $\mathcal{NP}$-Hard: The Subformula $\mathcal{F}_{accept}$

$\mathcal{F}_{accept}$ represents the requirement that the *bottom* row of the tableau represents an accepting configuration.

- Let $\mathcal{F}_{accept}$ be the formula

$$\bigvee_{1 \leq h \leq T(n)+2} x_{T(n),h,q_{accept}}$$

While this subformula is not sufficient by itself, one can see by inspection of its structure, and the requirements for the subformula $\mathcal{F}_{move}$, that if *this* subformula meets the requirements set for it, and

$$\mathcal{F}_{accept} \wedge \mathcal{F}_{move}$$

is satisfied under a truth assignment then the desired condition is met (because the bottom row represents a configuration), as required.

L_{FSAT} is $\mathcal{NP}$-Hard: The Subformula $\mathcal{F}_{accept}$

- Once again — because it is a very simple subformula — it is not difficult to argue that the length of $e(\mathcal{F}_{accept})$ is bounded by a polynomial function of $|\omega|$, and that the string $e(\mathcal{F}_{accept})$ can be computed deterministically from ω, using time bounded by a polynomial function of $|\omega|$, in this case too.

L_{FSAT} is $\mathcal{NP}$-Hard: The Subformula $\mathcal{F}_{move}$

$\mathcal{F}_{move}$ — the most complicated of these subformulas — represents the requirement that *every* row of the tableau represents a configuration and, furthermore, if $i \in \mathbb{N}$ and $0 \leq i \leq T(n)$, then either

- Rows i and $i+1$ represents configurations η_i and η_{i+1}, respectively, such that $\eta_i \vdash_M \eta_{i+1}$, or
- Row i represents a ***halting*** configuration η_i, and row $i+1$ represents η_i too.

A consideration of the above should confirm that the only things left, to do, are to describe the subformula $\mathcal{F}_{move}$, argue that it meets the conditions required for it, and that e($\mathcal{F}_{move}$) can be computed deterministically from ω, using a moves that is at most polynomial in $|\omega|$.

L_{FSAT} is $\mathcal{NP}$-Hard: The Subformula $\mathcal{F}_{move}$

- ***Windows*** will be used describe a subformula, $\mathcal{F}_{move}$, satisfying these conditions.
- ***Definition:*** For $0 \leq i \leq T(n) - 1$ and $0 \leq j \leq T(n) + 1$, the ***window*** $W_{i,j}$ is a condition identifying the contents of the cells of the tableau that are in rows i and $i + 1$ and columns j, $j + 1$ and $j + 2$ — that is, the contents of cells

$$C_{i,j}, C_{i,j+1}, C_{i,j+2}, C_{i+1,j}, C_{i+1,j+1}, C_{i+1,j+2}.$$

L_{FSAT} is $\mathcal{NP}$-Hard: The Subformula $\mathcal{F}_{move}$

- Windows will be shown pictorially. In particular, the condition that

$$C_{i,j} = \alpha_1, C_{i,j+1} = \alpha_2, C_{i,j+2} = \alpha_3, C_{i+1,j} = \beta_1,$$
$$C_{i+1,j+1} = \beta_2, \quad \text{and} \quad C_{i+1,j+2} = \beta_3,$$

  can be shown as a window $W_{i,j}$ that is drawn as follows:

α_1	α_2	α_3
β_1	β_2	β_3

- Each window can specify one of $(|Q| + |\Gamma| + 1)^6$ combinations of values for the contents of the cells it involves.

L_{FSAT} is $\mathcal{NP}$-Hard: The Subformula $\mathcal{F}_{move}$

- The condition (from the previous slide) that

$$C_{i,j} = \alpha_1, C_{i,j+1} = \alpha_2, C_{i,j+2} = \alpha_3, C_{i+1,j} = \beta_1,$$
$$C_{i+1,j+1} = \beta_2, \quad \text{and} \quad C_{i+1,j+2} = \beta_3,$$

  is (easily) represented by the subformula

$$(x_{i,j,\alpha_1} \wedge x_{i,j+1,\alpha_2} \wedge x_{i,j+2,\alpha_3} \wedge$$
$$x_{i+1,j,\beta_1} \wedge x_{i+1,j+1,\beta_1} \wedge x_{i+1,j+2,\beta_3})$$

L_{FSAT} is $\mathcal{NP}$-Hard: The Subformula $\mathcal{F}_{move}$

- A subset of the (combinations of values for) windows will be defined to be ***illegal*** because they can only arise in tableaux representing a sequence of configurations that do not correspond to valid moves of the Turing machine M.
- The other (combinations of values for) windows will be defined to be ***legal***.

L_{FSAT} is $\mathcal{NP}$-Hard: The Subformula $\mathcal{F}_{move}$

- Since there are only finitely many (combinations of values for) windows, there is a fixed set of ***legal*** (combinations of values for) windows. Thus a Boolean formula

$$\text{legal}_{i,j}$$

representing the claim that the (contents of values in) window $W_{i,j}$ is legal, is also easily described: It is the "disjunct," or ***or***, of a fixed number of subformulas that look like the above one, enclosed in parentheses.

L_{FSAT} is $\mathcal{NP}$-Hard: The Subformula $\mathcal{F}_{move}$

Nontrivial Facts:

- $\mathcal{F}_{move}$ can now be defined as the formula

$$\mathcal{F}_{move} = \bigwedge_{0 \leq i \leq T(n)-1} \left(\bigwedge_{0 \leq j \leq T(n)+1} \text{legal}_{i,j} \right). \qquad (2)$$

  It can be shown that this formula is satisfied if and only if the conditions associated with "$\mathcal{F}_{move}$" are met.
- The structure of this subformula is simple enough for it to be shown that an encoding of this subformula can be computed deterministically from ω using time polynomial in $|\omega|$.

These facts are, now, all that is needed to complete a proof that $L \preceq_{P,\,M} L_{FSAT}$, so that L_{FSAT} is $\mathcal{NP}$-hard.

L_{FSAT} is $\mathcal{NP}$-Hard: Defining $\text{legal}_{i,j}$

Consider the following *conditions*.

1. Every row begins and ends with # and there are no other copies of # in the row.
2. Every row includes exactly one cell whose contents is a *state*.
3. The change in state, contents of the tape cell initially visible, and change in tape head location when going from the configuration in one row to the configuration in the next row, is consistent with M's transition function — and all such changes, that are consistent with the transition function, are allowed.

L_{FSAT} is $\mathcal{NP}$-Hard: Defining $\text{legal}_{i,j}$

4. No symbol to the left of the position of the tape head is changed when going from the configuration in one row to the configuration in the row after that.
5. No symbol to the right of the position of the tape head is changed when going from the configuration in one row to the configuration in the row after that.

The first two conditions imply that every row of the tableau represents a configuration. The last three imply that, for $0 \leq i \leq T(n)$, if η_i is the configuration represented by row i and η_{i+1} is the configuration represented by row $i+1$, then either $\eta_i \vdash_M \eta_{i+1}$ or η_i is a *halting* configuration and $\eta_{i+1} = \eta_i$.

Thus they imply the conditions given for $\mathcal{F}_{\text{move}}$.

L_{FSAT} is $\mathcal{NP}$-Hard: Defining $\text{legal}_{i,j}$

- Consider the ***first*** requirement: Every row begins and ends with # and there are no other copies of # in the row.

  In order to satisfy this, let us declare that all windows such that

α_1	α_2	α_3
β_1	β_2	β_3

  such that either

  i. *exactly* one of α_1 or β_1 is $\#$, or
  ii. *exactly* one of α_3 or β_3 is $\#$, or
  iii. β_2 is $\#$

  are ***illegal***.

L_{FSAT} is $\mathcal{NP}$-Hard: Defining $\text{legal}_{i,j}$

Lemma #4: Let φ be any truth assignment such that

$$\varphi(\mathcal{F}_\omega) = \mathtt{T},$$

where $\mathcal{F}_\omega$ is as described above, and where the windows described in the previous slide are ***illegal***, as specified there.

Then φ specifies a well-defined assignment of values in $Q \cup \Gamma \cup \{\#\}$ to the cells in a two-dimensional table with $T(|\omega|) + 1$ rows and $T(|\omega|) + 4$ columns, such that the contents of each row begin and end with #, with no copy of # in between.

L_{FSAT} is $\mathcal{NP}$-Hard: Defining $\text{legal}_{i,j}$

How To Prove This:

- Note that, since $\varphi(\mathcal{F}_\omega) = \mathtt{T}$, $\varphi(\mathcal{F}_{\text{cell}}) = \mathtt{T}$ and it follows that φ specifies a well-defined assignment of values in $Q \cup \Gamma \cup \{\#\}$ to the cells in the two-dimensional table that is mentioned in the lemma.
- It remains only to prove that the following ***property*** is satisfied for every integer i such that $0 \leq i \leq T(|\omega|)$: *The contents of the i^{th} row of this table begin and end with #, with no copy of # in between.*

L_{FSAT} is $\mathcal{NP}$-Hard: Defining $\text{legal}_{i,j}$

- This can be proved by induction on i, using the standard form of mathematical induction — making use of the fact that $\varphi(\mathcal{F}_{\text{start}}) = \mathtt{T}$, $\varphi(\mathcal{F}_{\text{move}}) = \mathtt{T}$ and $\mathcal{F}_{\text{move}}$ is as shown at line (2), so that φ must specify values for table cells in such a way that all windows are legal. It suffices to notice which windows have been declared to be *illegal* after that.
- A complete proof is included in the supplement for this lecture. □

L_{FSAT} is $\mathcal{NP}$-Hard: Defining $\text{legal}_{i,j}$

- The construction — and proof of Claim #3— is completed by adding additional conditions about which windows are illegal (and which are illegal), for each of the remaining four conditions, and stating and proving claims, like the above lemma, establishing that these conditions on windows provide what is needed.
- The supplemental material includes the additional conditions on windows, lemmas, and proofs (or proof sketches) that are needed. □

Conclusions and Comments

Cook-Levin Theorem: L_{FSAT} is $\mathcal{NP}$-complete.

Proof: This is now a straightforward consequence of the definition of "$\mathcal{NP}$-complete" and the first two claims.

Cook's Conjecture: $\mathcal{P} \neq \mathcal{NP}$.

If Cook's conjecture is correct then no $\mathcal{NP}$-complete language, including L_{FSAT}, is in $\mathcal{P}$.

Final Note: The proof of the Cook-Levin Theorem outlined in Sipser's text, and given in these lecture notes, is the shortest and simplest proof of this result — that is not so vague that it is arguably not a proof, at all — that I know of.

Tableaux are also used to prove other results. We will (probably) see these again.