Lecture #11: The Cook-Levin Theorem
Lecture Presentation

Proving the Cook-Levin Theorem

Claim. *The language L_{FSAT} is $\mathcal{NP}$-complete.*

Why This Result is Significant:

How It Was Proved That $L_{\text{FSAT}} \in \mathcal{NP}$:

Proving That L_{FSAT} is $\mathcal{NP}$-Hard:

We must prove that $L \preceq_{\text{P, M}} L_{\text{FSAT}}$ for every language L such that $L \in \mathcal{NP}$ — so we begin with a language L, along with a nondeterministic Turing machine

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

and positive integers d, c_1 and c_0 such that M decides L and, furthermore, such that for every string $\omega \in \Sigma^\star$, the depth of the computation tree (of M) for ω is at most $T(|\omega|)$, for a function $T : \mathbb{N} \to \mathbb{N}$ such that $T(n) = c_1 n^d + c_0$ for every non-negative integer n.

What We Must Describe, at a High Level:

It is necessary to define a mapping from strings $\omega \in \Sigma^\star$ to Boolean formulas $\mathcal{F}_\omega$ such that $\omega \in L$ if and only $\mathcal{F}_\omega$, for every string $\omega \in \Sigma^\star$.

A Useful "Intermediate" Structure

It is assumed, now, that $c_0 \geq 4$ and that "#" is a symbol that does not belong to $Q \cup \Gamma$. (It is also assumed, as usual, that $Q \cap \Gamma = \emptyset$). The value of c_0 can certainly be increased, if needed, to ensure that $c_0 \geq 4$, and a replacement for the symbol "#" can be chosen, if this symbol belongs to $Q \cup \Gamma$. A ***tableau*** is two-dimensional table with $T(n) + 1$ rows and $T(n) + 4$ columns that satisfies the following properties.

(a) Each cell $C_{i,j}$ of the table (for $0 \leq i \leq T(n)$ and $0 \leq j \leq T(n) + 3$) 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.

(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.

(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.

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}. It follows that $\omega \in L$ if and only if there exists an ***accepting configuration*** for M and ω — and it is now sufficient to describe a construction of a Boolean formula $\mathcal{F}_\omega$ such that $\mathcal{F}_\omega$ is satisfiable if and only if $\mathcal{F}_\omega$ has an accepting tableau.

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 `T` under a truth assignment) if the symbol σ is stored in row i and column j of the tableau.

As discussed in the lecture notes there is a simple way to map these new variables to the variables in the set

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

that should be used when defining the Boolean formula $\mathcal{F}_\omega$.

$\mathcal{F}_\omega$ 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.
- $\mathcal{F}_{\text{move}}$ — 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.

Each of the subformulas $\mathcal{F}_{\text{cell}}$, $\mathcal{F}_{\text{start}}$ and $\mathcal{F}_{\text{accept}}$ have very simple structures — making it reasonably easy to see that they correctly represent what they are supposed to, and making reasonably easy to see that their encodings can be computed from the input string ω, using a number of steps that is at most polynomial in $|\omega|$.

Windows — as defined next — will be used to describe the remaining subformula, $\mathcal{F}_{\text{move}}$.

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

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

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

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

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 — and the above condition is (easily) represented by the subformula

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

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***.

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.

While it is *not* obvious, it can be shown that the subformula $\mathcal{F}_{\text{move}}$ can be defined as the formula

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

That is, it can be shown that this formula is satisfied if and only if the conditions associated with "$\mathcal{F}_{\text{move}}$" 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 $|\omega|$.

Using a Satisfying Truth Assignment to Construct an Accepting Tableau:

Using an Accepting Tableau to Construct a Satisfying Truth Assignment:

Other Formula Satisfiability Problems of Interest

Let $\mathcal{F}$ be a Boolean formula, defined using the set of Boolean variables

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

as described in the lecture notes.

Unsatisfiable Boolean Formulas

A Boolean formula $\mathcal{F}$ is ***unsatisfiable*** if $\varphi(\mathcal{F}) = \mathtt{F}$ for *every* partial truth assignment φ that is complete enough for a truth value of $\mathcal{F}$ to be defined, at all.

This suggests a "decision problem"' which might be called the ***Boolean Formula Unsatisfiability*** problem: Is a given Boolean formula, $\mathcal{F}$, *unsatisfiable*?

Let $L_{\text{FUnsat}} \subseteq L_{\text{F}}$ be the language of encodings of unsatisfiable Boolean formulas.

How is L_{FUnsat} Related to the Languages We Have Already Seen?

What Can Be Proved About the Complexity of L_{FUnsat}?

Notes about How to Prove This:

Falsifiable Boolean Formulas

A Boolean formula $\mathcal{F}$ is ***falsifiable*** if there exists a (partial) truth assignment φ such that $\varphi(\mathcal{F}) = \mathrm{F}$.

This suggests a "decision problem"' which might be called the ***Boolean Formula Falsifiability*** problem: Is a given Boolean formula, $\mathcal{F}$, *falsifiable*?

Let $L_{\text{Falsifiable}} \subseteq L_{\text{F}}$ be the language of encodings of falsifiable Boolean formulas.

How is $L_{\text{Falsifiable}}$ Related to the Languages We Have Already Seen?

What Can Be Proved About the Complexity of $L_{\text{Falsifiable}}$?

Notes about How to Prove This:

Tautologies

A Boolean formula $\mathcal{F}$ is a ***tautology*** if $\varphi(\mathcal{F}) = \mathtt{T}$ for *every* partial truth assignment φ that is complete enough for a truth value of $\mathcal{F}$ to be defined, at all.

This suggests a "decision problem"' which might be called the ***Tautology*** problem: Is a given Boolean formula, $\mathcal{F}$, a *tautology*?

Let $L_{\text{Tautology}} \subseteq L_{\text{F}}$ be the language of encodings of tautologies.

How is $L_{\text{Tautology}}$ Related to the Languages We Have Already Seen?

What Can Be Proved About the Complexity of $L_{\text{Tautology}}$?

Notes about How to Prove This: