An informal proof of the Cook-Levin Theorem

Douglas R. Stinson
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada

April 1, 2019

In this note, we give a very informal proof that a certain type of satisfiability problem is NP-complete. The problem is named Circuit-SAT and it is defined as follows. We consider boolean circuits that contain an arbitrary number of boolean inputs and one boolean output. The gates in a boolean circuit are and, or and not gates. An instance of Circuit-SAT consists of a boolean circuit \( C \) having inputs \( x_1, \ldots, x_n \), say. The question to be answered is if there is a truth assignment that assigns a boolean value (true or false) to each input, such that the output of the circuit \( C \) is true.

To prove from first principles that Circuit-SAT is NP-complete, we need to show two things:

1. Circuit-SAT \( \in \text{NP} \), and
2. for any \( \Pi \in \text{NP} \), there is a polynomial transformation from \( \Pi \) to Circuit-SAT.

It is fairly obvious that \( \text{Circuit-SAT} \in \text{NP} \) (a certificate for a yes-instance just consists of a satisfying truth assignment), so we concentrate on proving 2.

Suppose \( \Pi \) is in NP. Then there is a polynomial-time certificate verification algorithm for \( \Pi \), say \( \text{Ver}(I, \text{Cert}) \), where \( I \) is an instance of \( \Pi \) and Cert is a certificate. The running time of \( \text{Ver} \) is a polynomial function of size(\( I \)).

In a rigorous proof, the main work involved is to show that the algorithm \( \text{Ver} \) can be converted in polynomial time to a boolean circuit \( C \) that performs the same computation. Here we will just make the (plausible?) assumption that this conversion is possible. There is one technical detail that we need to address. A circuit has a fixed number of inputs, whereas an algorithm allows inputs of any size. So we are really assuming the existence of a family of circuits, one for each possible value of \( n = \text{size}(I) \).

Therefore, at this point, we assume we can construct (in polynomial time) a circuit \( C \) that evaluates \( \text{Ver}(I, \text{Cert}) \) for any instance \( I \) (of the decision problem \( \Pi \)) having any (fixed) size \( n \). We need to describe the polynomial transformation from \( \Pi \) to Circuit-SAT. That is, given an instance \( I \) of \( \Pi \), we need to show how to construct (in polynomial time) an instance \( f(I) \) of Circuit-SAT, such that \( f(I) \) is a yes-instance if and only if \( I \) is a yes-instance.

Given \( I \) with size(\( I \)) = \( n \), we construct \( f(I) \) in two steps:

1. Construct the circuit \( C \) that evaluates \( \text{Ver} \) for instances of \( \Pi \) having size \( n \).
2. Fix (i.e., hardwire) the boolean inputs to \( C \) that correspond to the given instance \( I \). Call the resulting circuit \( C' \) and define \( f(I) = C' \). Note that the values of the boolean inputs to \( C' \) that correspond to a certificate \( \text{Cert} \) are still unspecified.
We claim that the mapping $I \rightarrow f(I)$ is a polynomial transformation. First, it is clear that $f(I)$ can be computed in polynomial time. So we need to show that $f(I)$ is a yes-instance if and only if $I$ is a yes-instance.

Suppose that $I$ is a yes-instance of $\Pi$. Then there must exist a certificate $Cert$ such that $Ver(I, Cert) = true$. If we now use $Cert$ to set the inputs to $C'$, then $C'$ will evaluate to true.

Finally, suppose that $C' = f(I)$ is a yes-instance of Circuit-SAT. This means that there is a truth assignment for the inputs to $C'$ that cause $C'$ to output true. This truth assignment corresponds to a certificate $Cert$ such that $Ver(I, Cert) = true$. Therefore $I$ is a yes-instance of $\Pi$. 

\[
\begin{array}{c}
\text{I} \in \text{NP} \\
\text{Ver}_{\Pi}(I, Cert) \rightarrow \text{instance } f(I) \\
\text{I} \rightarrow (1) \rightarrow \text{I, Cert} \\
\text{certificates verifying circuit of the appropriate size} \\
\downarrow \text{fix inputs corresponding to } I \\
\text{resulting circuit} \\
\end{array}
\]