The Cook-Levin Theorem

Recall that a language \( L \) is \( \text{NP-complete} \) if \( L \in \text{NP} \) and if \( L \) is at least as hard as every language in \( \text{NP} \): for all \( A \in \text{NP} \), we have that \( A \leq_p L \). Our first \( \text{NP-complete} \) language is the hardest to get, since we have no \( \text{NP-hard} \) language to reduce to it. A first \( \text{NP-complete} \) language is provided by the Cook-Levin theorem, due to Stephen Cook (1971, USA/Canada) and, independently, Leonid Levin (1973, but the subject of lectures, in Russia, for some years before). The particular \( \text{NP-complete} \) problem we select is not of great importance; we will use SAT. What is more important is that we show some particular language \( \text{NP-complete} \) so, using it, we can start populating our universe with other known-to-be-\( \text{NP-complete} \) problems.

**Theorem [Cook-Levin].** SAT is \( \text{NP-complete} \).

To prove the theorem we must show that SAT \( \in \text{NP} \), which we know, and that, for any \( A \in \text{NP} \), we can poly-time reduce \( A \) to SAT. So fix \( A \in \text{NP} \), some \( \text{NP-complete} \) language. Fix \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R) \), a verifier that accepts \( A \). Fix \( p(n) \), a polynomial that upperbounds the running time of \( M \): the number of steps \( \text{TIME}_M(w \sqcup c) \) that \( M(w \sqcup c) \) takes is always less than \( p(n) \), where \( n = |w| \) and \( c \in \Gamma^* \) is arbitrary. We know that

- \( w \in A \Rightarrow (\exists c)M(w \sqcup c) \) accepts
- \( w \notin A \Rightarrow (\forall c)M(w \sqcup c) \) rejects

We haven’t been very explicit about where the certificate \( c \) is drawn from. We may consider it to be an element of \( \Gamma^* \). In fact, given our bound on the running time of \( A \), we may assume that \( c \in \Gamma^{p(n) - 1 - n} \). Strings longer than this will not even have their rightmost characters read.

Nor our job is to, by polynomial-time transformation, map \( w \in \Sigma^* \) to a Boolean formula \( \phi \) such that \( w \in A \) iff \( \phi \) is satisfiable. Our transformation will depend on machine \( M \) and polynomial \( p \). To describe \( \phi \), fix \( w \in \Sigma^* \). Let \( n = |w| \).

First, we specify the variables that \( \phi \) will use. These are

1. \( Q_{q,t} \) for each \( q \in Q \) and \( 1 \leq t \leq p(n) \).
   Variable \( Q_{q,t} \) is supposed to mean that machine \( M \) is in state \( q \) at time \( t \).

2. \( H_{i,t} \) for each \( 1 \leq i \leq p(n) \), \( 1 \leq t \leq p(n) \).
   Variable \( H_{i,t} \) is supposed to mean that the head of the machine \( M \) is at position \( i \) at time \( t \).

3. \( X_{a,i,t} \) for each \( a \in \Gamma \), \( 1 \leq i \leq p(n) \), \( 1 \leq t \leq p(n) \).
   Variable \( X_{a,i,t} \) is supposed to mean that there is an \( a \)-character at position \( i \) of the tape at time \( t \).

Now “all” we have to do is to write a collection of Boolean constraints that collectively capture the idea that our machine \( M \), on input \( w \sqcup c \) (for the given \( w \) and an arbitrary \( c \)), computes correctly and winds up in an accepting state. If you AND together all the constraints you get a Boolean formula that will be satisfiable iff \( w \in L \). Let’s show how some of these constraints look.
1. The machine starts off in its start state:
\[ Q_{q_0,1} \iff 1 \]

2. The head starts off at the left edge:
\[ H_{1,1} \iff 1 \]

3. The tape starts off with a \( w \sqcup c \) written on it:
\[
\begin{align*}
X_{w[i],i,1} & \iff 1 \text{ for all } 1 \leq i \leq n \\
X_{\sqcup,n+1,1} & \iff 1 \\
\bigvee_{a \in \Gamma} X_{a,i,1} & \iff 1 \text{ for each } n + 2 \leq i \leq p(n)
\end{align*}
\]

4. You end up in an accept state.
\[
\bigvee_{1 \leq t \leq p(n)} Q_{q_A,t}
\]

5. Each step of the machine is computed according to the transition. In particular, if \( \delta(q,a) = (q',b,R) \) then
\[
(Q_{q,t} \land H_{i,t} \land X_{a,i,t}) \Rightarrow (Q_{q',t+1} \land H_{i+1,t+1} \land X_{b,i,t+1}) \quad \text{for all } 1 \leq i < p(n), 1 \leq t < p(n)
\]
Similarly define the following constraints for when \( \delta(q,a) = (q',b,L) \). Here it is convenient to assume that \( M \) never tries to move its head to the left of the left edge of the tape, which is without loss of generality.
\[
(Q_{q,t} \land H_{i,t} \land X_{a,i,t}) \Rightarrow (Q_{q',t+1} \land H_{i-1,t+1} \land X_{b,i,t+1}) \quad \text{for all } 1 \leq i < p(n), 1 \leq t < p(n)
\]
Finally, if the head is not the immediate vicinity, the tape contents should simply be copied:
\[
(H_{i,t} \land X_{a,j,t}) \Rightarrow X_{a,i,t+1} \quad \text{for all } 1 \leq i,j < p(n), i \neq j, 1 \leq t < p(n)
\]

6. If you’re in one state, you’re not in another; if your head is somewhere, it’s not somewhere else; if something is written on a tape cell, nothing else isn’t written there.
\[
\begin{align*}
Q_{q,t} & \rightarrow \overline{Q_{q',t}} \quad \text{for all } q, q' \in Q, q \neq q', 1 \leq t \leq p(n) \\
H_{i,t} & \rightarrow \overline{H_{j,t}} \quad \text{for all } 1 \leq i,j \leq p(n), i \neq j, 1 \leq t \leq p(n) \\
X_{a,i,t} & \rightarrow \overline{X_{b,i,t}} \quad \text{for all } a,b \in \Gamma, a \neq b, 1 \leq i \leq p(n), 1 \leq t \leq p(n)
\end{align*}
\]

New we should verify the following: (1) The transformation is polynomial time. This is clear. Of course the polynomial depends on \( p(n) \), which depends on \( L \). That is as one would expect. (2) if \( w \in L(M) \) then \( \phi \) is satisfiable. This is easy; the computation of \( M \) on a certificate that demonstrates \( w \in L \) provides a satisfying assignment of \( \phi \). (3) if \( \phi \) is satisfiable, then \( w \in L(M) \). This is the most tricky part. We read the certificate \( c \) that demonstrates \( w \in L \) off of the satisfying assignment of \( \phi \). We have to have added enough constraints in our formula that a satisfying assignment really does correspond to possessing a certificate \( c \) and then performing a correct, accepting computation of \( M \) on input \( w \sqcup c \).