The Cook-Levin Theorem

This material was presented in class on January 26 and 28, 2016. Throughout, points that were not covered in detail in class and that you are encouraged to think through and justify are marked by "(WHY?)."

Recall that \( L_1 \) is polynomial-time, many-to-one reducible to \( L_2 \), denoted \( L_1 \leq_P L_2 \), if there is a polynomial-time computable function \( f \) such that \( x \in L_1 \) if and only if \( f(x) \in L_2 \). Reductions of this form are also referred to as Karp reductions. Throughout this lecture, all reductions are of this form, and we will refer to them simply as reductions.

The Cook-Levin Theorem tells us that, if \( L \in \text{NP} \), then \( L \leq_P \text{SAT} \). We will in fact prove that, if \( L \in \text{NP} \), then \( L \leq_P \text{3SAT} \).

**Lemma 1** \( \text{SAT} \leq_P \text{3SAT} \).

**Proof.** Let \( \phi = C_1 \land \cdots \land C_m \) be a SAT instance on the boolean variables \( x_1, \ldots, x_n \). For \( 1 \leq j \leq m \), \( C_j \) is the disjunction of literals \( \ell_{j,1}, \ldots, \ell_{j,k(j)} \), where each \( \ell_{j,q} \) is either \( x_r \) or \( \overline{x_r} \) for some \( r \), \( 1 \leq r \leq n \). Without loss of generality, we assume that \( k(j) \leq n \), for \( 1 \leq j \leq m \).

We give a recursive description of a reduction \( f \) that produces a formula that is in \( \text{3SAT} \) if and only if \( \phi \) is in SAT. In the base case of \( f \), all clauses in \( \phi \) have at most three literals; in this case, we just output \( \phi \) and halt. Otherwise, we proceed as follows.

Each clause \( C_j = \ell_{j,1} \lor \cdots \lor \ell_{j,k(j)} \) gives rise to either one or two clauses in a new formula \( \tau \). If \( k(j) \leq 3 \), then \( C_j \) is a clause in \( \tau \). Otherwise, let \( C'_j = \ell_{j,1} \lor \cdots \lor \ell_{j,s} \lor y_j \), and \( C''_j = \ell_{j,s+1} \lor \cdots \lor \ell_{j,k(j)} \lor \overline{y_j} \), where \( s = \lfloor k(j)/2 \rfloor \), and the set \( \{ y_1, \ldots, y_m \} \) of boolean variables is disjoint from \( \{ x_1, \ldots, x_n \} \). Both \( C'_j \) and \( C''_j \) are clauses in \( \tau \).

Now set \( \phi \) equal to \( \tau \) and make the next recursive call of \( f \).

We show that, in each recursive call, \( \tau \) is satisfiable if and only if \( \phi \) is satisfiable. Let \( A : \{ x_1, \ldots, x_n \} \rightarrow \{ T, F \} \) be an assignment of truth values to the variables of \( \phi \). Consider extending \( A \) to an assignment \( A' : \{ x_1, \ldots, x_n, y_1, \ldots, y_m \} \rightarrow \{ T, F \} \) of truth values to the variables of \( \tau \); \( A' \) is an "extension" in that it agrees with \( A \) on the variables of \( \phi \). If \( A \) satisfies \( \phi \), then any extension \( A' \) must satisfy \( C'_j \) or \( C''_j \) (or both), for \( 1 \leq j \leq m \). (WHY?) Assume without loss of generality that \( C'_j \) is satisfied regardless of whether \( A'(y_j) \) is \( T \) or \( F \); we can set the value of \( A'(y_j) \) so as to guarantee that \( C''_j \) is satisfied. This shows that \( \tau \) is satisfiable if \( \phi \) is satisfiable. If \( A \) falsifies \( \phi \), then it falsifies \( C_j \), for some \( j, 1 \leq j \leq m \). Thus all possible extensions yield assignments \( A' \) that falsify either \( C'_j \) or \( C''_j \). (WHY?). This shows that \( \tau \) is unsatisfiable if \( \phi \) is unsatisfiable.

Finally, note that \( f \) runs in time polynomial in \( n \) and \( m \). (WHY?).

We will say that the **size** of a CNF formula is the number of \( \lor \) and \( \land \) operators that it contains. This measure of size is polynomially related to the number of bits in any reasonable binary encoding of the formula.

**Proposition 2** The formula \( (x_1 \lor \overline{y_1}) \land (\overline{x_1} \lor y_1) \land \cdots \land (x_n \lor \overline{y_n}) \land (\overline{x_n} \lor y_n) \) is a CNF formula of size \( 4n - 1 = O(n) \) that is true if and only if \( x_i = y_i \), for \( 1 \leq i \leq n \).

**Lemma 3** For any function \( F : \{ 0, 1 \}^\ell \rightarrow \{ 0, 1 \} \), there is an \( \ell \)-variable CNF formula \( \phi_F \) of size at most \( \ell \cdot 2^\ell \) such that \( \phi_F(u) = F(u) \), for all \( u \in \{ 0, 1 \}^\ell \).
Proof. For each of the $2^\ell$ assignments $v \in \{0,1\}^\ell$, we may include in $\phi_F$ a clause $C_v$ that evaluates to 0 on $v$ and to 1 on all $u \neq v$. For example, if $v = (1,1,0,1)$, then $C_v(w_1, w_2, w_3, w_4) = \overline{w_1} \lor w_2 \lor w_3 \lor \overline{w_4}$. In general, if $v = (\epsilon_1, \epsilon_2, \ldots, \epsilon_\ell)$, then

$$C_v(w_1, w_2, \ldots, w_n) = \bigvee_{i=1}^\ell z_i,$$

where $z_i = w_i$ if $\epsilon_i = 0$, and $z_i = \overline{w_i}$ if $\epsilon_i = 1$. Note that all of the literals in this clause will be 0 if and only if the input is identical to $v$.

Now let

$$\phi_F = \bigwedge_{v \in F^{-1}(0)} C_v.$$

That is, the clause $C_v$ is included in the conjunction if and only if $F(v) = 0$. The size of $\phi_F$ is at most $\ell \cdot 2^\ell$, because there are at most $2^\ell$ vectors $v$ such that $F(v) = 0$, and each corresponding $C_v$ has size $\ell - 1$.

To see that $\phi_F(u) = F(u)$, note that, if $F(u) = 0$, then $C_u(u) = 0$, and thus the conjunction $\phi_F(u) = 0$, because it contains the clause $C_u(u)$. On the other hand, if $F(u) = 1$, then all clauses $C_v(u)$ in $\phi_F(u)$ evaluate to 1, because the clause is included only if $F(v) = 0$.

Note that, for a fixed constant $\ell$, $F$ corresponds to a constant-sized $\phi_F$.

We now proceed to the proof of the Cook-Levin Theorem. Let $L$ be a set in NP, and let $M$ be a (deterministic) polynomial-time TM and $p$ a polynomial such that

$$x \in L \iff \exists u \in \{0,1\}^{p(|x|)} \text{ s.t. } M(x,u) = 1.$$

Assume without loss of generality that $M$ has one input tape and one work/output tape. (Claim 1.6 in your textbook implies that this assumption is valid.) Assume further that $M$ is oblivious. By this, we mean that $M$’s tape-head movements do not depend on the contents of its input tape. Thus, the positions of $M$’s two tape heads at time $i$ depend on $i$ and on $|y|$ (the length of its input) but not on which string of length $|y|$ is on the input tape. (The fact that every polynomial-time function can be computed obliviously in polynomial time is something that you will work out in HW1.)

For any input length $m$, we can thus define inputpos$_m(i)$, which is the location of $M$’s input-tape head at time $i$, and prev$_m(i)$, which is the largest $j < i$ such that $M$’s work/output-tape head was at the same location at time $j$ as it is at time $i$. Note that, for each location in the work/output tape, there will be a smallest $i$ such that the work/output-tape head points to that location at time $i$. For that $i$, we set prev$_m(i) = 0$ and note that the symbol in that location at time 0 is $\square$.

Both inputpos$_m()$ and prev$_m()$ are computable in time polynomial in $m$. (WHY?).

Let $Q$ be the state set of $M$ and $\Gamma = \{\square, \triangleright, 0, 1\}$ be the tape alphabet of $M$. (Claim 1.5 in your textbook states that this four-symbol alphabet suffices.) A snapshot of $M$’s execution on input $y$ of length $m$ at time $i$ is a triple $z_i = \langle a, b, q \rangle \in \Gamma \times \Gamma \times Q$, where $a$ is the symbol read from $M$’s input tape at time $i$, $b$ is the symbol read from $M$’s work/output tape at time $i$, and...
$q$ is the state that $M$ is in at time $i$.

Note that the binary encoding of $z_i$ is of constant length $c$. (WHY?)

There is a function $G$ such that

$$z_i = G(z_{i-1}, z_{prev_m(i)}, y_{inputpos_m(i)}).$$

$G$ is determined by the transition function $\delta$ of the Turing Machine $M$. To compute the state in $z_i$, one needs merely to apply $\delta$ to $z_{i-1}$. The value $a$ in $z_i$ is, by definition, $y_{inputpos_m(i)}$. To compute the value $b$ in $z_i$, one needs precisely the information encoded by $z_{prev_m(i)}$, because it is the state, the input-tape symbol, and the work/output-tape symbol at time $j$ to which $M$ applies the function $\delta$ to determine which symbol $b$ to write on the work/output tape, and it is this symbol that will be read at time $i$. Note that $G : \{0,1\}^{2c+1} \rightarrow \{0,1\}^c$ is equivalent to a binary function $F : \{0,1\}^{3c+1} \rightarrow \{0,1\}$, to which we can apply Lemma 3: $F(z_1, z_2, z_3, a) = 1$ if and only if $z_1 = G(z_2, z_3, a)$.

Now we can put all of this together for our reduction from $x$ to $\phi_x$, such that $x \in L$ if and only if $\phi_x$ is satisfiable. Recall that $x \in \{0,1\}^n$ and $x \in L$ if and only if there is a $u \in \{0,1\}^{p(|x|)}$ such that $M(y) = 1$, where $y = xu$. So $|y| = m = n + p(n)$. Let $T(n)$ be the running time of $M$ on inputs of length $n + p(n)$. (Note that we need $M$ to be oblivious for $T(n)$ to be well defined.)

Consider the sequence of bits

$$y_1, \ldots, y_n, y_{n+1}, \ldots, y_{n+p(n)}, t_1, \ldots, t_{cT(n)},$$

where $xu = y_1, \ldots, y_{n+p(n)}$, $z_i = t_{(i-1)c+1}, \ldots, t_{ic}$ for $1 \leq i \leq T(n)$, and $z_1, \ldots, z_{T(n)}$ is the sequence of snapshots of $M$’s execution on $y$. Then $M(y) = 1$ if and only if this sequence of bits satisfies

1. The $x = y_1, y_2, \ldots, y_n$,
2. $z_1 = \langle \triangleright, \square, q_{start} \rangle$,
3. For all $i \in \{2, \ldots, T(n)\}$, $z_i = G(z_{i-1}, z_{prev_m(i)}, y_{inputpos_m(i)})$, and
4. $z_{T(n)}$ is a snapshot with $q = q_{HALT}$ and $b = 1$.

The output $\phi_x$ of our reduction is the formula on boolean variables $y_1, \ldots, y_{n+p(n)}, t_1, \ldots, t_{cT(n)}$ that is true if and only if the variables satisfy (1), (2), (3), and (4). (Note that $y_{n+1}, \ldots, y_{n+T(n)}$, i.e., the variables that encode the witness $u$, are the only ones that must be chosen in order satisfy the formula; the others are determined by $x$, $u$, and $M$.) Moreover, (1) gives rise to a formula of size $O(n)$, by Proposition 2; (2) and (4) are formulae of size $O(1)$; and (3) is the AND of $T(n) - 1$ formulae of size $O(1)$ by Lemma 3. Thus $\phi_x$ can be computed in polynomial time, given $x$ and $M$. 
