In this worksheet, we outline a proof that SAT is NP-Complete (known as the **Cook-Levin Theorem**) (from Wikipedia). This proof has two parts.

1. SAT $\in$ NP.
2. SAT is *NP-hard*: Any language $L \in$ NP will satisfy $L \leq_p$ SAT.

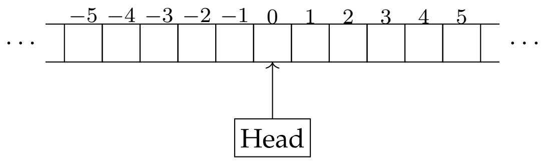
Exercise 1. Show that SAT $\in$ NP. That is, build a poly-time nondeterministic Turing machine that decides SAT.

SAT is NP-hard

Let $L \in$ NP. We will show that $L \leq_p$ SAT by constructing a poly-time computable function f such that $x \in L \Leftrightarrow f(x) \in$ SAT.

Since $L \in$ NP, there is a poly-time nondeterministic Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ that decides L. Since M is poly-time, we may assume M halts in $\leq p(n)$ steps on any input of size n (where $p(n)$ is some polynomial). We will make use of the following lemma:

Lemma. Suppose $M(x)$ finishes executing in $p(n)$ steps. Then during execution, only cells $-p(n)$ to $p(n)$ could have been accessed during execution; cells $-\infty$ to $-p(n) - 1$, and cells $p(n) + 1$ to ∞ are inaccessible to the read/write head.



Exercise 2. Briefly justify the Lemma. *Hint: How long does it take to move the read/write head?*

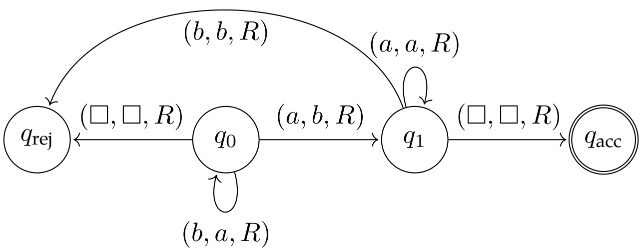
The Lemma guarantees that if M is given an input x of size n, then throughout $M(x)$'s execution, only cells $-p(n)$ to $p(n)$ can be overwritten.

Now, given an input x, we will define the following collections of variables:

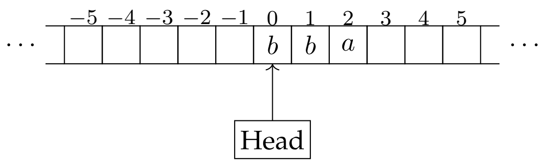
Variables	Range	Intended Interpretation
$T_{i,j,k}$	$-p(n) \leq i \leq p(n)$, $j \in \Sigma$, $0 \leq k \leq p(n)$	Cell i has symbol j at step k of execution.
$H_{i,k}$	$-p(n) \leq i \leq p(n)$, $0 \leq k \leq p(n)$	The read/write head is at cell i at step k of execution.
$Q_{q,k}$	$q \in Q$, $0 \leq k \leq p(n)$	The NTM is at state q at step k of execution.

Table 1: Variables and their intended interpretations

Exercise 3. Suppose M is the *deterministic* Turing machine below (with $\Sigma = \{a, b\}$). You may assume that $M(x)$ halts in $p(n) = n + 1$ steps or less.



Execute M on the input $x = bba$ (so M halts in $p(n) = 3 + 1 = 4$ steps), and label the variables below with their intended interpretations (consulting Table 1).



Variable	Value
$T_{0,a,0}$	F
$T_{1,b,0}$	T
$T_{2,b,0}$	F
$T_{0,a,1}$	
$T_{1,a,1}$	
$T_{2,a,2}$	

Variable	Value
$T_{3,\square,2}$	
$T_{-2,\square,2}$	
$T_{2,b,2}$	
$T_{0,a,3}$	
$T_{1,a,4}$	
$T_{2,a,4}$	

Variable	Value
$H_{0,0}$	T
$H_{1,0}$	F
$H_{1,1}$	T
$H_{2,1}$	
$H_{2,2}$	
$H_{3,3}$	

Variable	Value
$Q_{q_0,0}$	T
$Q_{q_1,0}$	F
$Q_{q_1,1}$	F
$Q_{q_1,2}$	
$Q_{q_1,3}$	
$Q_{q_{\text{acc}},4}$	

In Table 1, there are:

- $(2p(n)+1) \cdot |\Sigma| \cdot (p(n)+1) = \mathcal{O}(p(n)^2)$ variables of the form $T_{i,j,k}$.
- $(2p(n)+1) \cdot (p(n)+1) = \mathcal{O}(p(n)^2)$ variables of the form $H_{i,k}$.
- $|Q| \cdot (p(n)+1) = \mathcal{O}(p(n))$ variables of the form $Q_{q,k}$.

In total, we have created $\mathcal{O}(p(n)^2)$ variables given an input x of size n.

Recall that we want to create a poly-time computable f so that $x \in L \Leftrightarrow f(x) \in \text{SAT}$. In other words, given an x, we want to create a boolean formula $f(x)$ in polynomial time, so that $f(x)$ is satisfiable iff $x \in L$. Here's how we create this boolean formula $f(x)$:

- The variables of this boolean formula are the $T_{i,j,k}$'s, the $H_{i,k}$'s, and the $Q_{q,k}$'s, as defined in Table 1. There are $\mathcal{O}(p(n)^2)$ variables.
- The formula is the *conjunction* ($\wedge$) of all of the following boolean subformulae (with $-p(n) \le i \le p(n)$, $0 \le k \le p(n)$):

Formulae	Range	Interpretation	How many formulae?
$T_{i,j,0}$	$j \in \Sigma$ cell i initially contains symbol j	Initial contents of the tape	$\mathcal{O}(p(n))$
$Q_{q_0,0}$		TM starts in state q_0	1
$H_{0,0}$		R/W head starts at cell 0	1
$\neg T_{i,j,k} \vee \neg T_{i,j',k}$	$j, j' \in \Sigma$ with $j \neq j'$	At most 1 symbol per cell	$\mathcal{O}(p(n)^2)$
$\bigvee_{j \in \Sigma} T_{i,j,k}$		At least 1 symbol per cell	$\mathcal{O}(p(n)^2)$
$T_{i,j,k} \wedge T_{i,j',k+1} \to H_{i,k}$	$j, j' \in \Sigma$ with $j \neq j'$	To change cell i, head must be at cell i	$\mathcal{O}(p(n)^2)$
$\neg Q_{q,k} \vee \neg Q_{q',k}$	$q, q' \in Q$ with $q \neq q'$	Only one state at a time	$\mathcal{O}(p(n))$
$\neg H_{i,k} \vee \neg H_{i',k}$	$i \neq i'$	Only one head position at a time	$\mathcal{O}(p(n)^3)$
$(H_{i,k} \wedge Q_{q,k} \wedge T_{i,j,k}) \to \bigvee_{(q,j),(q',j',d) \in \delta} \begin{pmatrix} H_{i+d,k+1} \\ \wedge Q_{q',k+1} \\ \wedge T_{i,j',k+1} \end{pmatrix}$	$j \in \Sigma$ $q \in Q$ $k \neq p(n)$	Non-deterministic transition function δ is obeyed	$\mathcal{O}(p(n)^2)$
$\bigvee_{0 \le k \le p(n)} Q_{q_{\text{acc}},k}$		Accepting state q_{acc} reached within $p(n)$ steps	1

Table 2: $f(x)$ is the conjunction ($\wedge$) of all of the following subformulae.

Notice that this huge boolean formula $f(x)$ is satisfiable iff $x \in L$:

- If $f(x)$ is satisfiable, then there is some execution path in $M(x)$ that ends in the accepting state q_{acc}. It follows from the definition (since M is a nondeterministic decider for L) that $x \in L$.
- If $x \in L$, then there is some execution path in $M(x)$ leading to acceptance. We may then assign the $T_{i,j,k}$'s, the $H_{i,k}$'s, and the $Q_{q,k}$'s according to their intended interpretation to satisfy $f(x)$.

$f(x)$ takes $\mathcal{O}(p(n)^3)$ time to produce, and since $p(n)$ is polynomial, so is $p(n)^3$. Thus f is indeed poly-time computable. This shows $L \le_p \text{SAT}$. $\square$

Exercise 4. Referring back to the Turing machine M in Exercise 3, M accepts the input $x = bba$.

(a) List the variables in $f(x)$ (consulting Table 1).

(b) Write down the boolean formula $f(x)$ (consulting table 2).