Alex Hubers
CSC:341 Spring 2024

Lab #12A

Due: April 27, 2024

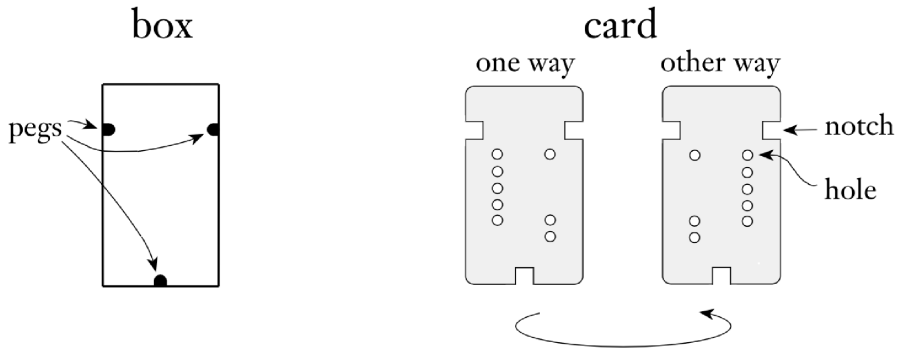
#1

Answer the following key questions about the proof of the Cook-Levine theorem that tests our understanding of Turing machines and polynomial-time reductions.

1. What are the inputs and outputs of the mapping function of the reduction?
2. Why do the tableau under consideration have n^k rows?
3. Why can we bound the size of the tape, i.e., the number of columns of the tableau, to n^k?
4. What do the boolean variables $x_{i,j,s}$ represent in the reduction?
5. In a few sentences each, describe the structure and purpose of the following boolean formulae:
   - ϕ_{cell}
   - ϕ_{start}
   - ϕ_{move}
   - ϕ_{accept}
6. Does a 2×2 window, i.e., two rows and two columns, work? Why or why not?
7. How does the constructed boolean function account for the nondeterminism of the NTM it simulates?

#2

You are given a box and a collection of cards as indicated in the following figure. Because of the pegs in the box and the notches in the cards, each card will fit in the box in either of two ways. Each card contains two columns of holes, some of which may not be punched out. The puzzle is solved by placing all the cards in the box so as to completely cover the bottom of the box (every hole position is blocked by at least one card that has no hole there).



Define a problem, called PUZZLE, below.

$\mathsf{PUZZLE} = \{\langle c_1, \ldots, c_k \rangle \mid$ Each c_i represents a card and this collection of cards has a solution $\}$.

Ultimately, our goal will be to show that PUZZLE is NP-complete. The following steps lead us through the proof, however you may or may not find each step useful in your thought process. The only required parts (that you must turn in) are part b and part f.

1. First, let's explore this problem space a little bit. Design two sets of cards, C_1 and C_2. For C_1, ensure that there is a solution to the puzzle, i.e., an orientation of each card such that the bottom of the box is covered, and list this solution. For C_2, ensure that there is no such solution to the puzzle. Try to make these example cards non-degenerate, (for example, don't choose every card to have no holes) so that you can use your examples to generalize these results.

2. To show NP-completeness, we must show both conditions of Definition 7.34. First, show that PUZZLE $\in$ NP.

   (Hint: what is a potential solution to PUZZLE? How do we verify that the solution is correct?)

3. Now, let's consider the second condition of Definition 7.34. (Note: If a language satisfies the second condition, we call the language NP-hard.) Here, we will show NP-hardness by reducing SAT to PUZZLE where we know via the reading that SAT is NP-complete. First, let's identify the "inputs" to each of the problems. What are the inputs to (predicates that decide) SAT and PUZZLE?

4. With the inputs, let's constrain our reduction function f. Write down the type of the reduction function f, i.e., what does f take as input and produce as output?

5. Next we must design a the function f. Let's experiment with designs that link together these parts and see if we can strike gold! To do this productively, first design a toy SAT example of 3–4 variables and 3–4 clauses with a satisfying assignment. Give at least two potential designs for reductions from SAT to PUZZLE showing how your reductions operate on your toy example, even if the designs do not work! If the designs do not work, describe what was broken about the reduction and how you could then address it in a subsequent attempt.

6. Repeat the previous step until you find a working reduction. Once you have this reduction, formally show that SAT $\leq_p$ PUZZLE, demonstrating that PUZZLE is NP-hard. Make sure to prove both sides of the correctness condition of your reduction!