Problem Set 10

Jack Schneiderhan & Jessica Noel

Problem 1

Show that the language $L = \{ \langle M, w, k \rangle :$ TM M accepts input w and never moves its head beyond the first k tape cells $\}$ is decidable.

The language above is able to provided decidable by:

- Construct a TM M_1 marking with a $\$$ the cell k cells away from with the initial head cell.
- Simulate the input w over the machine M.
- If the input is accepted before the $\$$ is reached, M ACCEPTS.
- Otherwise M REJECTS

Because of this, the language is decidable.

Problem 2

Recall that $A \leq_p B$ if there is a polynomial time computable function f such $w \in A \leftrightarrow f(w) \in B$.

a) Show that the relation $\leq_p$ over languages is transitive

- Assume $A \leq_p B$ and $B \leq_p C$. Therefore $A \leq_p C$.
- Let f be a polynomial function.

  reducibility function over $A \& B$. and g be a reducibility function over $B \& C$.
- If $w \in A \leftrightarrow g(f(w)) \in B$

  Therefore $g(f(w))$ is polynomial time computable.
- If we then bound $|f(w)|$ by a polynomial, this also binds $h(w)$ as $n(w)$ is reliant on $f(w)$ in this case.
- From this, $w \in A \quad f(w) \in B$

  $w \in B \leftrightarrow \quad g(w) \in C$

  $w \in A \leftrightarrow g(f(w)) \in C$
- Relating this to our initial condition, if $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$. From this definition $\leq_p$ is transitive.

Continuation of part 2

b) Show that if $\forall A, B \in P$, if $B \neq \emptyset$ and $B \neq \Sigma^*$ then $A \leq_p B$.

- Given: $A \in P$
- Let f be a polynomial time computable function
- From this, $f(w) = a_0$ if $w \in A$ and $f(w) = a_1$ if $w \notin A$
  - ↳ Let $a_0 \in B$ and $a_1 \in B^c$
- If $w \in A$ iff $f(w) \in B$, (proven since the previous statement) then $A \leq_p B$, proving the initial

c) Show that if P = statement. NP, then every language other than $\emptyset$ & Σ^*, in P is NP-complete.

- from part(b), since $B \in P$, we can say $a_0 \in B$ and $a_1 \notin B$.
- Let f be a polynomial time computable function from a language L to B.
- $f(w) = a_0$ if $w \in L$ and $f(w) = a_1$ if $w \notin L$
- This proves that there is a polynomial-time computable function from L to B
- This proves that B is NP-complete.
- If P = NP, then every other language other than $\emptyset$ & Σ^* is NP-complete. (where P in this case is B), proving the initial statement.

Problem 3

a) Starting with n variables (x_1, x_2, x_n) and the formula from the genie if $x_1 \wedge$ (formula) is satisfiable. If he says no, then you know that x_1 itself must be false. However, if he says the formula is true the x_1 must be true. From there, not whether x_1 is T/F from your previous test, and continue with the next inquiry: whether $x_2 \wedge$ (formula) is satisfiable or not. Proceed with this process until you have answers for each variable, and then you have your truth assignments for each variable $(x_1, x_2, \ldots, x_n)$

b) This algorithm can have a max number of n queries

c) A satisfying assignment is an assignment to all variables such that the formula itself evaluates to true. This algorithm does exactly that by finding the correct truth assignments in order to evaluate the formula to be deemed satisfiable.

d) For the second genie, start with the undirected graph. Start with edge 1, and ask the genie to remove this edge, then ask if there is still a Hamiltonian Cycle. If there is, leave the edge omitted as that shows the edge does not "play a part" in the Hamiltonian cycle. If there is no longer a H. cycle, then add the edge back. Repeat this process for the entire graph, and at the end you will discover the Hamiltonian cycle in the graph.

Problem 4

a) Give a high-level description of a linear time algorithm to determine if a directed graph contains a directed cycle.

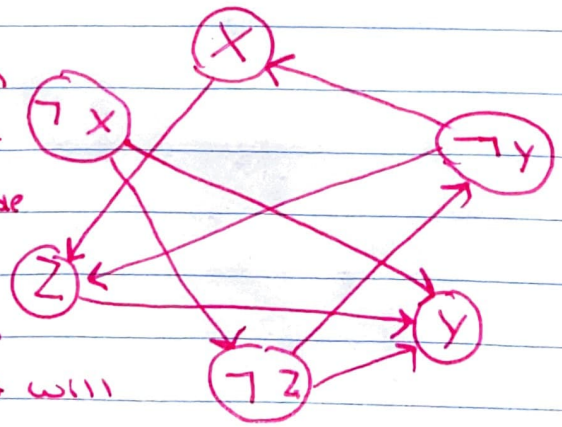
To determine whether or not a directed graph contains a cycle, we can utilize DFS or BFS:

- BFS approach:
  - Allow the BFS to traverse through the graph. Each time a node is reached, add the node to a queue.
  - If a node gets added to the queue twice, then this shows there is a cycle within the graph.
- DFS approach:
  - Allow the DFS to traverse through the graph. Each time a node is reached, add the node to a stack.
  - If a node gets added to the stack twice, then this shows there is a cycle within the graph.
- The differences between stack/queue are important as they both matter to the speed of each respective algorithm. If a node is never added twice, then there is no cycle. These algorithms take a time of $O(V+E)$ time, where V is # of vertices and E is # of edges.

b) $(x_i \vee x_j)$, add $(\neg x_i, x_j)$ and $(\neg x_j, x_i)$

$(\neg x_i \vee x_j)$, add (x_i, x_j) and $(\neg x_j, \neg x_i)$

Let CNF $= (x \vee y) \wedge (y \vee z) \wedge (\neg x \vee z) \wedge (\neg z \vee y)$



The BFS/DFS algorithm can determine whether or not this 2CNF is satisfiable by checking for paths between certain variables. If the path exists, it will show if the corresponding expression in the CNF exists. BFS/DFS can search for these edges to return if the overall CNF is/is not satisfiable.