Nondeterministic Turing Machines and Cook's Theorem

- Turing machine (TM)
  - Memory is a 1-way infinite tape
    - Each tape cell has a finite capacity (holds one object of type char)
    - There is one tape head, and the only cell that can be read is the one the tape head is at
  - Instructions
    - The only instructions available to a TM are reading a character, writing a character, moving the tape head one cell to the left or right, and goto instructions
  - Despite the simplicity of this model, a TM is a general model of computation
    - Church-Turing Thesis: Any algorithm can be expressed as a TM
    - Extended Church-Turing Thesis: Any polynomial-time algorithm can be expressed as a TM that operates in polynomial time
      - There is ample evidence to support both claims, but no proof is possible without destroying the spirit of the claims
      - The extended thesis needs to be adapted to handle probabilistic algorithms and quantum algorithms

- Nondeterminism
  - In most algorithmic models, the next step is always determined
    - A computation can be expressed as a path (sequence of configurations)
  - In a nondeterministic algorithm, multiple next steps are possible
    - A computation now is expressed as a tree or more generally as a directed acyclic graph
  - Example: If $(a == b)$ goto (7, 9, or 11) else goto (13 or 15)
  - Acceptance/rejection of an input by a nondeterministic algorithm
    - A nondeterministic algorithm A accepts an input if there exists an accepting path within its computation graph for that input.
    - A nondeterministic algorithm A rejects an input if all paths within its computation graph for that input are rejecting.
  - Measuring time for a nondeterministic algorithm
    - A nondeterministic algorithm A operates in nondeterministic polynomial time if the HEIGHT of its computation tree for all inputs of size n is bounded by a polynomial in n.

- **NP**
  - The set of decision problems that are solved by some nondeterministic algorithm (TM) that operates in nondeterministic polynomial-time is the class **NP**
  - Connection to previous definition of **NP**
    * One way to think about a nondeterministic algorithm
      · First guess a polynomial-sized certificate $C(I)$ nondeterministically
      · Then apply deterministic verification algorithm to all guessed certificates
    * Another way to think about the certificate $C(I)$
      · A certificate can be the sequence of choices made by the NTM in accepting input I
      · Given the certificate and the NTM, we can verify that I is accepted by the NTM in deterministic polynomial time.

Theorem 1 *Let L be a language decided by a deterministic Turing Machine $M = (K, \Sigma, \delta, s)$ which operates within time $f(n)$. Then there exists a deterministic Turing Machine M' which, for any string x, constructs an instance I of CIRCUIT VALUE with the following properties:*

1. *$|I| \leq (f(n))^k$ for some constant k that depends only on M*
2. *M' produces I within time $f(n)^{k'}$ for some constant k' that depends only on M*
3. *M accepts x if and only if I evaluates to TRUE.*

Proof:

- Define the *table* $T(M, x)$ of M's computation on x as follows
  - $T(M, x)$ has $f(n) + 1$ rows numbered 0 through $f(n)$
  - $T(M, x)$ has $f(n)$ columns numbered 1 through $f(n)$
  - The entry in the i^{th} row and j^{th} column is labeled $T(M, x)[i, j]$.
  - $T(M, x)[i, j]$ contains two fields as follows:
    * Field one of $T(M, x)[i, j]$ corresponds to the contents of the j^{th} cell of M's work tape after time step i for $0 \leq i \leq f(n)$ and $1 \leq j \leq f(n)$.
    * Field two of $T(M, x)[i, j]$ takes one of $|K| + 1$ different values depending on the location of the tape head of M after the i^{th} time step of M on input x.
      · If the tape head is in cell j, this field is set to the state of M after the i^{th} time step of M on input x.
      · If the tape head is not in cell j, this field is set to a null state value.
  - Note M may halt on x in time step $t < f(n)$. In this case, all subsequent rows duplicate row t.
  - $T(M, x)$ includes *all* the relevant information of M's computation on x.

- How large must each table entry be?
  - The first field of each table entry can assume at most $|\Sigma|$ distinct values.
  - The second field of each table entry can assume at most $|K| + 1$ distinct values.
  - Thus each table entry must have size $\log |\Sigma|(|K|+1)$ which is a constant dependent only on M.
- What information can be used to compute $T(M, x)[i, j]$ for $i > 0$?
  - $T(M, x)[i, j]$ depends only on $T(M, x)[i-1, j-1]$, $T(M, x)[i-1, j]$, and $T(M, x)[i-1, j+1]$.
  - In particular, if the tape head of M is not in cells $j-1$ through $j+1$ in time step $i-1$, $T(M, x)[i, j] = T(M, x)[i-1, j]$.
- How can $T(M, x)[i, j]$ for $i > 0$ be computed?
  - Note $T(M, x)[i, j]$ depends on $T(M, x)[i-1, j-1]$, $T(M, x)[i-1, j]$, and $T(M, x)[i-1, j+1]$ in exactly the same manner for all $i > 0$ and $1 < j < f(n)$ (ignoring side columns).
  - Thus a single Boolean circuit C can be used to compute $T(M, x)[i, j]$ from $T(M, x)[i-1, j-1]$, $T(M, x)[i-1, j]$, and $T(M, x)[i-1, j+1]$ for these values of i and j.
    - Need to handle "side table entries" with a slightly modified circuit.
  - The size of C is a constant dependent only on M.
- Costructing the instance (circuit) I
  - The input gates to the circuit I corresponds to the initial configuration of M's computation on x; that is row 0 of table $T(M, x)$
    - Easily computed from input x
  - Most of the rest of circuit I is circuit C (or the slightly modified C to handle side conditions) replicated many times
    - How many copies of circuit C do we need?
  - Output value
    - Final tree of OR gates that OR together all the field 2's of the last row of the table to see if M is in a yes state
  - I conforms to conditions we demanded
    - What is the size of I?
    - Why are we guaranteed that I is a yes input to CIRCUIT VALUE if and only if x is a yes input to L?

□

Corollary 1 *CIRCUIT SAT is* **NP***-complete.*

Lemma 1 *CIRCUIT SAT* $\leq_p$ *SAT*