Top Hat

What kind of machine (or machine type) can verify computation by a (possibly nondeterministic, L-tape) Turing Machine \( M = (Q, \Sigma, \Gamma, \delta, q_0, F) \) on an input \( x \)?

Computation \( c = [I_0(x)] [I_1] [I_2] \ldots [I_j] [I_{j+1}] \ldots [I_t] \)

Want to be a halting TD.

Prototypical of checking a proof.

Checking \( I_j \downarrow \) and \( I_{j+1} \) is mostly like checking equality of two strings (over \( \Sigma^* \)).

If we write odd \( I_j \), \( \Sigma \).

Except we need to check \( I_{j+1} \) follows from a legal \( I_j \).

in reverse, \( \Sigma \).

Checking a legal instruction in \( \Sigma \).

This is a very local edit.

marked by \[ \] brackets. \( c = [I_0(x)] [I_1 R] [I_2] [I_3 R] [I_4] [I_5] \ldots [I_t] \)

A demo of two DPDA's:

\( D_1 \) and \( D_2 \).

Push \( D_1 \).

Do pop \( D_2 \).

Check for \( u \) in \( \Sigma^* \).

A TM M is halting.

A demo of two DPDA's: (on the \( \Sigma \) in finite control. DPDA \( D_1 \), \( D_2 \).)

whichever boolean check whether last \( D_2 \).

is halting.

The language \( VC_M \) of valid accepting computation by a TM \( M \) is the intersection of two DCFLs.

The complement \( \overline{VC_M} \) is a CFL. Check failure of \( I_j \downarrow \) \( I_{j+1} \) in one place and checking \( I_j \downarrow \quad I_{j+1} R \).

(Or with \( I_k \).)

is like the complement of palindromes which is a CFL; with grammar \( G \).

\( \text{M \in E_{TM} \iff \overline{VC_M} = \emptyset} \iff \overline{VC_M} = \Sigma^* \iff L(D_1) \cap L(D_2) = \emptyset \)

\( \overline{L(D_1)} \cup \overline{L(D_2)} = \emptyset \)

\( \text{M has no valid accepting computation on any input } x. \)

This is the correctness condition for reductions from \( E_{TM} \) to these two problems.
Theorem: A DTM or N TM runs in polynomial time if for all \( x \in \Sigma^* \),
\[
N \leq 3|L_{TM}| = M \quad \text{runs in polynomial time on all inputs.
}\]

Hence, the classical solution to the problem of finding the shortest path
in the graph is to use a polynomial-time algorithm.

Proof: Every algorithm that runs in \( \mathcal{O}(n^e) \) time on \( n \) steps,
where \( e \leq 4 \), is in the complexity class \( \mathcal{P} \).

Linear Bounded Automata (LBAs) are capable of recognizing
\( \mathcal{L} \) when only \( \mathcal{O}(n) \) tape
space is available. The LBAs can be

Two-Head DFAs

Two-Head DFAs are capable of recognizing
\( \mathcal{L} \) when only \( \mathcal{O}(n) \) tape
space is available. The LBAs can be

Linear Bounded Automata (LBAs) are capable of recognizing
\( \mathcal{L} \) when only \( \mathcal{O}(n) \) tape
space is available. The LBAs can be

Linear Bounded Automata (LBAs) are capable of recognizing
\( \mathcal{L} \) when only \( \mathcal{O}(n) \) tape
space is available. The LBAs can be
Thm (Ch3): For every NTM N we can build a DTM M such that LN = L(M).

Proof: N simulated by Java, which runs in polynomial time. We construct a RAM simulator for M, maintaining a data structure of all computation branches, looping over 1-step updates of each one until you find that some branch accepts.

Problem: M will still take exponential (nl) time from this.

Central Question: Can we do it faster? \( P = NP? \)

Theorem: A language \( L \) belongs to NP if and only if there is a polynomial time decidable language \( R \) in \( P \) such that for all \( x \in \Sigma^* \), \( x \in L \iff \exists y : |y| \leq p(|x|) \cdot R(x,y) \).

Proof: Take \( N \) to \( N \) accepts \( L \) and runs in poly time \( g(n) \).

Then \( x \in L \iff (\exists y : c \text{ has an acc comp of } N \text{ on input } x \text{ with poly-time verifier}) \).

Verify this with a 2-HDFA, which runs in \( O(|c|) \) time, and \( |c| \leq g(n)^2 \).

Since \( [c] \leq g(n) \) steps time, max size of any IP i in c.

Added: Conversely, given a poly time verifier \( M_2 \) for \( R(x,y) \), we can build an NTM \( N \) that on input \( x \) guesses \( y \) and then verifies \( R(x,y) \).

Since \( |x| \leq p_1(|x|) \) and \( \text{poly}(\text{poly}(n)) = \text{poly}(n) \), \( N \) runs in poly time, so \( L \in NP \).

The Ch4 deciders for DFA, NFA, ALL DFA, E DFA, ECFG all run in poly time.