6.1400 Cheat Sheet (Computability & Complexity)

Gatlen Culp (gculp@mit.edu) · 2025-05-21

1 Legend

`Monospace` = Turing Machines
Sans-serif = Languages
CAPS = Language Classes

2 Regular Languages

Closure under:

$$A \cup B, \quad A \cap B, \quad A^R, \quad A^*, \quad \neg A,$$
$$A \cdot B, \quad A - B = A \cap (\neg B)$$

Order of Operations: $* \to \cdot \to +$

3 DFA

$$\text{DFA} = (Q, \Sigma, \delta, q_0, F)$$

3.1 DFA → MinDFA

3.1.1 Defining MinDFA and Ideas

Note: The minimum DFA is unique

A **MinDFA** requires (1) no indistinguishable states, (2) no unreachable states

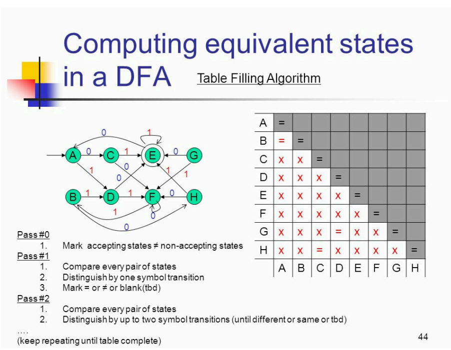
$$\mathtt{M}_p := \mathtt{M} \text{ but starting on state } P$$

$$w \in \Sigma^* \text{ distinguishes states } p, q$$
$$:= \mathtt{M}_p \text{ and } \mathtt{M}_q \text{ have different outputs on } w$$

$$p \sim q := \text{states } p, q \text{ are distinguishable}$$
$$:= \exists w \text{ distinguishing } p, q$$

Distinguishable states form a partitions of equivalence on Q. These can be simplified after removing unreachable states.

3.1.2 Table-Filling Algorithm (DFA → MinDFA)



4 NFA

$$\text{NFA} = (Q, \Sigma, \delta, Q_0, F)$$

4.1 NFA → DFA

Create a table with n + 1 columns – one for the DFA state and one for each of n transitions. List out the transitions in each column. Add rows for each new possible set of states reachable.

State	a	B
q0	{q0,q1}	q0
{q0,q1}	{q0,q1}	{q0,q2}
{q0,q2}	{q0,q1}	q0

- Note: No bijection between $\{1, \ldots, n\}$ and $\underbrace{2^{\{1,\ldots,n\}}}_{\text{powerset}}$

4.2 NFA → GNFA → RegExp

```
if |states| = 2:
    return R(q_start, q_acc)
G' ← G.copy()
pick q_rip ∈ (Q − {q_start, q_acc})
Q' ← Q \ q_rip
∀q_i, q_j ∈ (Q' \ q_acc, Q' \ q_start):
    R'(q_i, q_j) ← [ R(q_i, q_rip) · R(q_rip, q_rip)^* · R(q_rip, q_j) ]   (R(q_rip, q_rip)^* : Self-loops)
                    + R(q_i, q_j)
Repeat from the top
```

$$R'(q_i, q_j) \leftarrow \left[R(q_i, q_{\text{rip}}) \cdot \underbrace{R(q_{\text{rip}}, q_{\text{rip}})^*}_{\text{Self-loops}} \cdot R(q_{\text{rip}}, q_j)\right] + R(q_i, q_j)$$

5 Streaming Algorithms

```
Initialize vars and their assignments on vars
When next symbol = σ:
    Run pseudocode for σ
Accept/reject based on vars
```

6 Myhill-Nerode Theorem

$\forall L \in$ Langs : either
$\exists$ DFA recognizing L
or
There are infinite strings to trick DFA attempting to recognize L

ie: equiv classes of $\underset{L}{\equiv}$ is finite $\Leftrightarrow L$ is regular

6.1 Distinguishing Set

Strings $w_1, \ldots, w_n, \underbrace{\ldots}_{\text{possibly infinite}}$ s.t. $\forall i \neq j$:

$$\exists z \text{ s.t. } (w_i z \in L) \text{ xor } (w_j z \in L)$$

Ex: For $L = \{0^n 1^n \mid n \geq 0\}$
take $S = \{0, 00, \ldots 0^n, \ldots\}$ where $z = 1^i$
Then $\forall w_i, w_j \in S : \quad [(0^i 1^i) \in L] \wedge [(0^j 0^i) \notin L]$

6.2 Streaming Distinguisher

Streaming Distinguisher D_n: Distinguishing set when limiting the concatenated strings to length n.

$$(\forall \text{ distinct } x, y \in D_n)(\exists \text{ word } z)$$
$$(xz \in L) \text{ xor } (yz \in L)$$
$$\wedge (|xz|, |yz| \leq n)$$

If $\forall n, \exists D_n$ s.t. $|D_n| \geq 2^{S(n)}$
then all streaming algs must also use $\geq s(n)$

7 Communication Complexity

7.1 Protocol

$$A, B : \Sigma^* \times (\Sigma^* \cup \text{STOP})$$

Which collectively compute f

If $\exists$ streaming alg using $\leq S(m)$ space on $2m$ inputs then:
$$\text{cc}(f) \leq O(s(m))$$

7.2 Switcheroo Lemma

(x, y) and (x', y') share comms pattern P,
then so do (x, y') and (x', y)

7.2.1 Example: ALICE-HAS-MORE Lower Bound

For ALICE-HAS-MORE(A, B), create set $S = ((1^{i+1}0^{n-i-1}, 1^i 0^{n-i}) : 0 \leq i \leq n-1)$.

All pairs in S have ALICE-HAS-MORE = 1.

If protocol uses $< \log_2 n$ bits, by pigeonhole, two pairs share same pattern: $(1^{i+1}0^{n-i-1}, 1^i 0^{n-i})$ and $(1^{j+1}0^{n-j-1}, 1^j 0^{n-j})$.

By switcheroo: $(1^{i+1}0^{n-i-1}, 1^j 0^{n-j})$ and $(1^{j+1}0^{n-j-1}, 1^i 0^{n-i})$ also share same pattern.

This implies $i + 1 > j$ and $j + 1 > i$, contradiction when $i \neq j$.

8 Turing Machines

$$(\mathtt{M} \in \mathtt{TMs}) = (Q, \Sigma, \Gamma, \delta, q_o, q_{\text{acc}}, q_{\text{rej}})$$

(1) Read σ (2) Write σ (3) Move left/right (4) Change states

9 Equivalence Classes

$$\left(x \underset{L}{\equiv} y\right) := (\forall z \in \Sigma^*)[xz \in L \Leftrightarrow yz \in L]$$
$$\Leftrightarrow x, y \text{ are } L \text{ equivalent}$$
$$\Leftrightarrow x, y \text{ are distinguishable to } L$$

This is similar to equivalence relation of DFA states. "Further transitions are identical."

Note: Equivalence classes require (1) Symmetric, (2) Transitive, (3) Reflexive

9.1 Under TMs

$$\Delta_{\mathtt{M}} : \Sigma^x \to Q$$
$$:= \text{State you end up in after reading a string}$$

$$x \underset{\mathtt{M}}{\approx} y \Leftrightarrow \Delta(x) = \Delta(y)$$
$$\Leftrightarrow x \underset{L}{\equiv} y$$
$$\Leftrightarrow x \text{ and } y \text{ are distinguishable strings under } \mathtt{M}$$

Where I'm pretty sure `M` is the machine recognizing L

10 Formal Systems & Their Limits

Formal system F

- finite language, notion of **proof**, notion of **truth**

"Interesting" iff

1. **Expressive** – each TM–input pair (M, w) maps (computably) to a sentence $S_{M,w}$ with $S_{M,w}$ **true** $\Rightarrow M$ accepts w.
2. **Checkable** – the set $\{(S, P) \mid P \textit{ is a proof of } S \textbf{ in } F\}$ is **decidable**.
3. **Halting-decisive** – if M halts on w, then F proves either $S_{M,w}$ or $\neg S_{M,w}$.

Consistency – no statement S with both S and $\neg S$ provable.
Completeness – every statement S has either S or $\neg S$ provable.

Limit theorems

- **Gödel (1931)**: any consistent, interesting F is **incomplete** (true but unprovable statements exist).
- **Gödel (1931)**: F cannot prove its own consistency.
- **Church–Turing (1936)**: deciding whether a sentence of F has a proof is **undecidable**.

11 Recursion Theorem

WLOG, a program can reference its its own code

$$\forall t : \Sigma^* \times \Sigma^* \to \Sigma^*$$
(Write t as if it already has recursion)
$$\exists R : \Sigma^* \to \Sigma^* \text{ s.t. } \forall R(w) = t(\langle R \rangle, w)$$

12 Turing Machine Minimization

(UNDECIDABLE)

Where $\mathtt{M} \in \mathtt{TMs}$:
$$\text{Minimal}(\langle M \rangle) \Leftrightarrow M \text{ is a minimal state TM}$$

13 Fixed Point Theorem

If $t : \Sigma^* \to \Sigma^*$ is computable:
$\exists \mathtt{F} \in \mathtt{TMs}$ s.t.
$t(\langle \mathtt{F} \rangle)$ outputs $\langle \mathtt{G} \rangle$ where $L(\mathtt{F}) = L(\mathtt{G})$

ie: We can make a copy that is behaviorally identical to the input for any string-to-string function t.

14 Rice's Theorem

"Program analysis is hard"

let $P :=$ Property
$: \mathtt{TMs} \to \{0, 1\}$
$\wedge P$ is Non-Trivial (Not all acc/rej)
$\wedge P$ is Semantic $(L(\mathtt{M}) = L(\mathtt{M'}) \Rightarrow P(\mathtt{M}) = P(\mathtt{M'}))$
then $\{\langle M \rangle \mid P(\mathtt{M}) = 1\} \in$ UNDECIDABLE

15 Decidable Predicates

$$R \in \text{Predicates}, \quad \mathtt{M} \in \mathtt{TMs}$$
$$R(x, y) \text{ is True} \Leftrightarrow \mathtt{M}(x, y) \text{ accepts}$$

Vaguely:
`NTMs` = RECOGNIZABLE
`TMs` = DECIDABLE.

16 Computational Complexity

$T(n) :=$ Time complexity of $\mathtt{M} \in \mathtt{TMs}$ on strings up to length n

$$\text{TIME}(t(n)) := \{L \mid L \text{ has time complexity } O(t(n))\}$$
$$\text{NTIME}(t(n)) := \{L \mid L \text{ is decided by } O(t(n)) \text{ time complexity } \mathtt{NTM}\}$$
$$P := \text{Poly Time} = \cup_{k \in \mathbb{N}} \text{TIME}(n^k)$$
$$\text{NP} := \text{Non-Deterministic Poly Time} = \cup_{k \in \mathbb{N}} \text{NTIME}(n^k)$$

17 Universal Turing Machines

$(\mathtt{U} \in \mathtt{UTMs}) = \mathtt{Universal\ TM}$
$\mathtt{U}$: Code × Input → Output
$\mathtt{U}$ on $\langle$tm, x$\rangle$, accept if ($\mathtt{M}$ accepts x) else reject

(A) Input tape, (B) State Tape, (C) Code Tape, (D) Simulation Tape

18 Time Hierarchy Theorem

"You can solve strictly more problems /w more time"

$$\textbf{if } g(n) > n^2 f(n)^2$$
$$\textbf{then } \text{TIME}(f(n)) \subset \text{TIME}(g(n))$$

19 Extended Church Turing Thesis

All intuitions about efficient algs = Poly-time `TMs`

20 Mapping Reduction

$$A \underset{m}{\leq} B := (\exists f : \Sigma^* \to E^*)$$
$$\text{s.t. } w \in A \Leftrightarrow f(w) \in B$$

20.1 Poly-Time Reduction $\leq_p$

Same as mapping reduce, but in poly-time (uses partial order)

21 NP-HARD and NP-COMPLETE

$$\text{NP-HARD} := \{L \mid \forall A \in \text{NP}, A \leq_p L\}$$

$$\text{NP-COMPLETE} := \{L \mid L \in \text{NP} \land$$
$$L \in \text{NP-HARD ie: not EXPTIME or smthn}\}$$

22 Non-Deterministic TMs (`NTMs`)

- May proceed according to several possible transitions
- Accepts if there is any accepting condition

23 Verifier Characterization of NP

$$L \in \text{NP}$$
$$\Leftrightarrow$$
$$\exists k \in \mathbb{R}, V(x, y) \in \{\text{Poly-time predicates}\} \text{ s.t.}$$
$$L = \{x \mid \exists y \in \Sigma^*[|y| \leq k|x|^k \land V(x, y)]\}$$

ie: Guess and check with some poly-time verifier

something something

$$V \text{ ran in poly}(n) \text{ time} \Leftrightarrow L \in \text{TIME}(2^{O(k^n)} \cdot \text{poly}(n))$$

24 Oracles

$X^Y :=$ Problems
solvable using any TM in X with an
using an oracle for the language Y
(or selecting any oracle from Y)

25 BPP & Error-Reduction

$$\text{BPP} := \left\{L \mid L \text{ is decided by a} \right.$$
$$\left. \text{poly-time probabilistic TM with error} \leq \frac{1}{3}\right\}$$

Any constant $< \frac{1}{2}$ works: once the error is bounded away from $\frac{1}{2}$ by $\frac{1}{\text{poly}(n)}$ we can **amplify**.

Error-reduction lemma Let $\varepsilon \in \left(0, \frac{1}{2}\right)$ and $k \in \mathbb{N}$. If machine M_1 runs in $t(n)$ and

$$\Pr(M_1(x) \neq L(x)) \leq \tfrac{1}{2} - \varepsilon,$$

there exists M_2 with

$$\Pr(M_2(x) \neq L(x)) < \frac{1}{2^{n^k}},$$
$$\text{time}(M_2) = O\left(n^k \cdot \frac{t(n)}{\varepsilon^2}\right).$$

Construction: run M_1 independently $m := \Theta\left(\frac{n^k}{\varepsilon^2}\right)$ times and output the majority vote.

Chernoff bound (additive form) For independent $\frac{0}{1}$ X_i, let $X = \sum_i X_i$, $\mu = E[X]$:

$$\Pr(X < (1 - \delta)\mu) \leq \exp\left(-\delta^2 \frac{\mu}{2}\right).$$

With $X_i = 1$ iff M_1 is correct $\Rightarrow \mu \geq \left(\frac{1}{2} + \varepsilon\right)m$. Choose $\delta = 2\frac{\varepsilon}{1+2\varepsilon}$ to obtain

$$\Pr(\text{majority wrong}) \leq \exp(-2\varepsilon^2 m)$$
$$< 2^{-\varepsilon^2 \frac{m}{\ln} 2}.$$

Setting $m = 10\frac{n^k}{\varepsilon^2}$ yields error $< 2^{-n^k}$.

A *monotone clause* in a CNF formula is a clause consisting of all positive literals or all negative literals. For example, $(x_1 \lor x_2 \lor x_3)$ and $(\neg x_1 \lor \neg x_2 \lor \neg x_3)$ are monotone clauses, but $(x_1 \lor \neg x_2 \lor \neg x_3)$ is not. The *clause-monotone SAT problem* is to decide the satisfiability of a CNF formula that consists only of monotone clauses.
Prove that the clause-monotone problem is NP-complete.

The clause-monotone SAT problem is clearly in NP, so it remains to prove that it is NP-hard. We reduce from 3SAT as follows:

Let $\phi = C_1 \land \cdots \land C_m$ be a 3CNF formula on variables $x_1, \ldots, x_n$.

We introduce a new variable y_i for each x_i. For each $C_i = (l_1 \lor l_2 \lor l_3)$, we let the clause D_i be $(m_1 \lor m_2 \lor m_3)$, where $m_i = x_j$ if $l_i = x_j$, and $m_i = y_j$ if $l_i = \neg x_j$.

Take $\phi' = D_1 \land \cdots \land D_m \land (x_1 \lor y_1) \land (\neg x_1 \lor \neg y_1) \land \cdots \land (x_n \lor y_n) \land (\neg x_n \lor \neg y_n)$.

$\phi \mapsto \phi'$ is a computable in polynomial time.

Note that any satisfying assignment for ϕ' is also a satisfying assignment for ϕ, and that any satisfying assignment for ϕ can be extended to a satisfying assignment for ϕ' by setting $y_i = \neg x_i$. In particular, ϕ is satisfiable iff ϕ' is satisfiable.

26 Problems

PRIMES $:= \{n \mid n \text{ is prime}\}$
in P

PATH $:= \{(G, s, t) \mid \text{Graph } G \text{ has a path from } s \text{ to } t\}$
in P

SAT $:= \{\phi \mid \phi \text{ is a satisfiable boolean formula}\}$
in NP-COMPLETE

3SAT $:= \{\phi \mid \phi \text{ is a satisfiable 3-CNF formula}\}$
in NP-COMPLETE

CIRCUIT-SAT $:= \{C \mid \text{Boolean circuit } C \text{ is satisfiable}\}$
in NP-HARD

HALT $:= \{\langle\text{Tm(M), w}\rangle \mid \texttt{M} \text{ halts on input } w\}$
in RECOGNIZABLE but UNDECIDABLE

NHALT $:= \{\langle\text{Tm(M), w}\rangle \mid \texttt{M} \text{ does not halt on input } w\}$
in UNRECOGNIZABLE

CLIQUE $:= \{(G, k) \mid \text{Graph } G \text{ has a clique of size} \geq k\}$
in NP-COMPLETE

MAX-CLIQUE $:= \{(G, k) \mid \text{Max clique size in graph } G \text{ is exactly } k\}$
in P^{NP}

IS $:= \{(G, k) \mid \text{Graph } G \text{ has an independent set of size} \geq k\}$
in NP-COMPLETE

VERTEX-COVER $:= \{(G, k) \mid \text{Graph } G \text{ has a vertex cover of size} \leq k\}$
in NP-COMPLETE

SUBSET-SUM $:= \left\{(S, t) \mid \exists S' \subset S \text{ s.t.} \sum_{x \in S'} x = t\right\}$
in NP-COMPLETE

KNAPSACK $:= \{(I, v, w, W, V) \mid \text{Item set } I \text{ with values } v \text{ and weights } w\}$
has subset with weight $\leq W$ and value $\geq V$
in NP-COMPLETE

HOM $:= \{(G, H) \mid \text{Exists homomorphism from graph } G \text{ to graph } H\}$
in NP-COMPLETE

HOM-EDGE $:= \{(G, H) \mid \text{Exists edge-preserving homomorphism from } G \text{ to } H\}$
in P

PARTITION $:= \left\{S \mid \exists S' \subset S \text{ s.t.} \sum_{x \in S'} x = \sum_{x \in S - S'} x\right\}$
in NP-COMPLETE

BIN-PACKING $:= \{(S, k, B) \mid \text{Items } S \text{ can fit in } k \text{ bins of capacity } B\}$
in NP-COMPLETE

SHORTEST-PATH $:= \{(G, s, t, k) \mid \text{Graph } G \text{ has path from } s \text{ to } t \text{ of length} \leq k\}$
in P

LONGEST-PATH $:= \{(G, s, t, k) \mid \text{Graph } G \text{ has path from } s \text{ to } t \text{ of length} \geq k\}$
in NP-COMPLETE

HAM-PATH $:= \{(G, s, t) \mid \text{Graph } G \text{ has Hamiltonian path from } s \text{ to } t\}$
in NP-COMPLETE

NO-HAMPATH $:= \{(G, s, t) \mid \text{Graph } G \text{ has no Hamiltonian path from } s \text{ to } t\}$
in coNP-COMPLETE

HAM-CYCLE $:= \{G \mid \text{Graph } G \text{ has a Hamiltonian cycle}\}$
in NP-COMPLETE

LOW-WEIGHT-2SAT $:= \{(\phi, k) \mid \text{2-CNF formula } \phi \text{ has satisfying assignment with <= k true variables}\}$
in NP-COMPLETE

MIN-WEIGHT-2SAT $:= \{(\phi, k) \mid \text{Min weight of satisfying assignment for 2-CNF formula } \phi \text{ is } k\}$
in P^{NP}

+-1-LI $:= \{(A, b) \mid \text{System } Ax = b \text{ has a solution with entries in } \{-1, 0, 1\}\}$
in NP-COMPLETE

TAUT = TAUTOLOGY $:= \{\phi \mid \phi \text{ is a tautology}\}$
in coNP-COMPLETE

UNSAT $:= \{\phi \mid \phi \text{ is unsatisfiable}\}$
in coNP-COMPLETE

FACTORING $:= \{(n, k) \mid n \text{ has a non-trivial factor} \leq k\}$
in NP $\cap$ coNP

FIRST-SAT $:= \{\phi \mid \text{First assignment in lex order satisfies } \phi\}$
in P^NP-COMPLETE

MIN-FORMULA $:= \{(\phi, k) \mid \phi \text{ has equivalent formula of size} \leq k\}$
in coNP^NP-COMPLETE

MIN-CNF $:= \{(\phi, k) \mid \phi \text{ has equivalent CNF formula of size} \leq k\}$
in coNP^NP-COMPLETE

TQBF $:= \{\phi \mid \text{Quantified boolean formula } \phi \text{ is true}\}$
in PSPACE-COMPLETE

GG $:= \{\text{Generalized Geography game}\}$
in PSPACE-COMPLETE

FG $:= \{\text{Formula Game}\}$
in PSPACE-COMPLETE

NEQUIV $:= \{(M_1, M_2) \mid M_1 \neq M_2\}$
in NP-COMPLETE

EQUIV $:= \{(M_1, M_2) \mid M_1 = M_2\}$
in coNP-COMPLETE

A_{DFA} $:= \{\langle M\rangle \mid M \text{ is a DFA that accepts some string}\}$
in DECIDABLE

A_{NFA} $:= \{\langle M\rangle \mid M \text{ is an NFA that accepts some string}\}$
in DECIDABLE

EQ_{DFA} $:= \{\langle\text{M\_1, M\_2}\rangle \mid M_1, M_2 \text{ are DFAs and } M_1 = M_2\}$
in DECIDABLE

EQ_{NFA} $:= \{\langle\text{M\_1, M\_2}\rangle \mid M_1, M_2 \text{ are NFAs and } M_1 = M_2\}$
in DECIDABLE

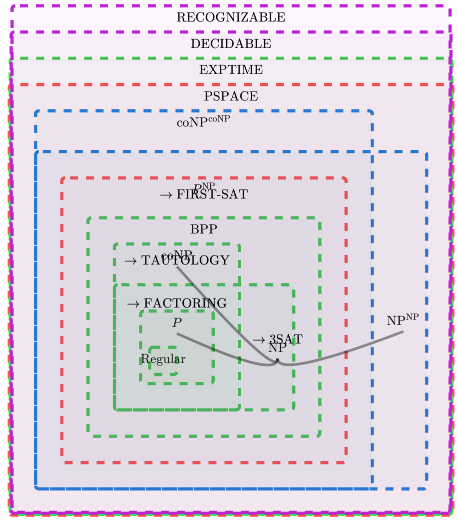
A_{TM} $:= \{\langle M\rangle \mid M \text{ is a TM that accepts some string}\}$
in RECOGNIZABLE but UNDECIDABLE

HALT_{TM} $:= \{\langle M\rangle \mid M \text{ is a TM that halts on some input}\}$
in RECOGNIZABLE

EMPTY_{TM} $:= \{\langle M\rangle \mid M \text{ is a TM and } M = \{\}\}$
in UNRECOGNIZABLE

EQ_{TM} $:= \{\langle\text{M\_1, M\_2}\rangle \mid M_1, M_2 \text{ are TMs and } M_1 = M_2\}$
in UNRECOGNIZABLE

27 Complexity Classes



Let $EMPTY_{TM} = \{\langle M\rangle \mid M \text{ is a Turing machine and } L(M) \text{ is empty}\}$.

(a) Prove that $EMPTY_{TM}$ is decidable with A_{TM}: that is, there is a Turing reduction from $EMPTY_{TM}$ to A_{TM}.

PROOF: First, we give a decider for $EMPTY_{TM}$ using an oracle for A_{TM}:

Let M be a given TM. Let M' be the following TM:
"On input x, for all pairs (y, z) in lex order, do:
If M reaches an accept state on y in at most $|z|$ steps, then accept."
If $((M', \varepsilon) \in A_{TM})$ then *reject, else accept.*

Why does this work? Observe that $L(M) \neq \emptyset$ if and only if there is some input y and some t such that, if M is run on y for t steps, then M accepts. This condition is true if and only if M' accepts ε (by definition of M'). And that is true if and only if $(M', \varepsilon) \in A_{TM}$. So the above decides correctly if $L(M) = \emptyset$ or not. ☐

(b) Prove that there is no mapping reduction from $EMPTY_{TM}$ to A_{TM}.

PROOF: If there were such a reduction, then $EMPTY_{TM}$ would be recognizable because A_{TM} is recognizable. However, the complement of $EMPTY_{TM}$ is also recognizable, because it can be written in the form:

$$\{M \mid (\exists y)(\exists t)[M \text{ reaches an accept state on input } y \text{ in at most } t \text{ steps }]\},$$

where the [...] is a decidable predicate. That would mean $EMPTY_{TM}$ is decidable, which we have shown is not true in lecture. (You could also prove it's undecidable by Rice's theorem.) ☐

(a) Let M be a deterministic Turing machine that, on inputs of length n, uses space $O(n^2)$. Then for every input x, and every configuration C of the computation of M on input x, $K(C) \leq O(|x|)$ where K is Kolmogorov complexity and $|x|$ is the length of x.

The claim is false.

PROOF: Let M be the following TM:
On input x:
Count to $2^{|x|^2}$ (in binary).
Reject.

First note that for any word w and any prefix v of w, $K(v) \in O(K(w) + \log |v|)$. Now let w be a string of length $|x|^2 - 1$. Since w is a prefix of some configuration C of M on input $|x|$, $K(w) \in O(K(C) + \log |x|)$. There exist incompressible strings of every length, so there is some configuration C such that $K(C) \in \Omega(|x|^2)$. ☐

(b) Let M be a deterministic Turing machine that, on inputs of length n, uses space $O(n^2)$ and time $O(n^3)$. Then for every input x, and every configuration C of the computation of M on input x, $K(C) \leq O(|x|)$, where K is Kolmogorov complexity and $|x|$ is the length of x.

The claim is true.

PROOF: Let M' be the following TM:
On input M, x, i:
Simulate M on input x for i steps.
Output the simulated configuration of M.

Let C be a configuration of M on input x. Then since M takes time $O(n^3)$, C is the ith configuration for some i represented as a binary string of length $O(\log |x|)$. Then since $|M|$ and $|M'|$ are fixed, $(M', (M, x, i))$ is a representation of C of length $O(|x|)$. ☐

28 Complexity/Computability Relationships

Given five languages with these properties:

A : in P
B : in NP
C : is NP-complete
D : is decidable
E : is recognizable but not decidable

Determining whether the following reductions are ALWAYS, MAYBE, or NEVER true:

$E \underset{T}{\leq} D$: NEVER true
$B \underset{p}{\leq} C$: ALWAYS true
$A \underset{p}{\leq} B$: MAY BE true
$B \underset{p}{\leq} \neg C$: MAY BE true
$D \underset{m}{\leq} C$: ALWAYS true