Principle of Information & Security

Siddharth Bhat

Contents

Chapter 1

Problems, Solutions, and Resources

1.1 Problems

Alphabet set is finite, call it Σ. Strings must be finite length.

Given some input, and a computer that produces some output, the description could be infinite — both input and output.

However, the machine's *description* (aka, the relationship between input and output) must be finite.

So, the *total input* can be infinite, but the input chunk must be finite, and the response of the machine per *input chunk* must be finite.

So, we can just use the language $L = \{0, 1\}$ for the machine.

Problems which have yes/no as answers are called decision problems. Inputs are from Σ^*, outputs are from $\{0, 1\}$. The problem is a mapping $f : \Sigma^* \to \{0, 1\}$. This is equivalent to providing the set $\texttt{ACCEPT} \subset \Sigma^* = f^{-1}(1)$. Note that $\texttt{REJECT} = \texttt{ACCEPT}^c$. The set `ACCEPT` is called a language.

Now, we can study languages by looking at their grammars (welcome, Chomsky).

What about fractional bit problems? Is this useful? Could we exploit some properties of fractional dimension?

Cantor set

take $S_0 = [0, 1]$ In each iteration, remove the middle one-third of each continuous interval. Therefore,

- $S_0 = [0, 1]$
- $S_1 = [0, \frac{1}{3}] \cup [\frac{2}{3}, 1]$

In S_∞, *uncountably infinite* points remain (However, this set has *measure* 0).

So now, the question is, what is the dimension? We define Haussdorf dimension, and use this to exhibit fractional dimension of the Cantor set.

`TODO: fill this up!`

The total number of problems that can exist is $powerset(\Sigma^*)$. `RE` (recursively enumerable) Is a subset of $powerset(\Sigma^*)$ which computers can handle. The annoying thing is that there are *finite length problems* which computers cannot solve.

1.1.1 Kannan

If the universe is a machine, then it must have infinite description.

QM is the meeting point of universes?

1.2 Solutions

1.2.1 Kannan

Question: We study a lot of Science — why? What is the ultimate goal of science? Equivalently, what is the theory of everything we need to find to halt on the journey of Science?

Assuming Science = God, we need to ask Science a question. Which language will we use to query Science? Or, well, which language is *enough* to query Science? If the query alphabet is Σ, we can ask Σ^* questions. However, we can only reasonably pose questions of finite length (even though the Science oracle can answer questions of infinite length).

In this case, have we achieved te ultimate goal of science?

1.3 Resources

Chapter 2

Diagonalization

- Level 1: $\mathbb{R}$ is uncountable.
- Level 2: $\exists L, L \notin \texttt{RE}$.
- Level 3: Halting problem is undecidable.
- Level 4: Time/Space hierarchy.
- Limitations: Exists oracles A, B such that — $P^A = NP^A, P^B \neq NP^B$
- Level 5: If $P \neq NP$, $\exists L, L \notin P, L \notin NPC$ (Ladner's theorem)

Diagonalization cannot separate P, NP — If it could, then it should also separate P with any oracle, and NP with the same oracle. We know that there exists an oracle such that we can separate $P^A = NP^A$, as well as $P^B \neq NP^B$.

chapterReview of the last 3 lectures, after add-drop

- Is it easier to *pose* problems than to *solve* them?
- Can every "solvable" problem have a solution that uses finite resources?
- What problems are *interesting*?
- Are all interesting problems solved in an *interesting* way? (P v/s NP)
- Can things get more interesting? (Quantum Mechanics, Approximation, Randomness, Interactivitiy, ...)

Are there problems with infinite length input / output but can still be posed in finite time? Eg. output π in decimal. However, we decided that both input/output should be finite. We decided this does not belong to problems we wish to solve it, since we cannot solve it in finite time. If we believe that nature is inherently noisy, or nature is quantized, or nature has finite precision, then we cannot consider problems that require infinite time as problems in this universe (since Nature / the universe itself cannot pose such a problem).

Quantum mechanics (which is a theory of quantization) is developed over infinite precision mathematics ($\mathbb{C}$). Does this really make sense? There is a way in which a quantized universe can be infinite precision: This is by using 'external help': There are infinite such quantized universes which intersect at some points, and at those points, precision will increase. (If we both have a resolution of 1 pixel but are at a gap of 1/2, my least count is now 1/2). If there are an infinite number of universes overlapping at a single point, then we can construct "infinite precision". (*I feel this is crazy. Is this really crazy?*)

Posing a question is creating a language $L \subset \Sigma^*$. (Sid: a solution is a classifier for L).

Kannan's view:

- Finite space $\equiv$ finite information can be stored. (Turing: finite tape alphabet. Since a cell demarcates a finite volume, we want to have a finite amount of info in this cell)
- Information travels at finite speed. If we have cells, we should not be able to store and retreive information "equally" (based on how far we are from it). Hence, all infinite memory must be sequential memory since information travels at finite speed.
- Finite program $\equiv$ finite control.

Solution to these choices is a TM.

Do all languages have a TM recognizing it? No ($\mathtt{RE}$ = solvable by TM).

The class $\mathtt{R}$ = decidable by a TM (TM halts on all inputs). Diagonalization led us to Halting problem.

We have the class P, and we claimed that P is interesting. Given that P is considered interesting because of feasibility, it is possible that there are questions that are interesting even though **solving them** is not feasible. For example, if we can actually **understand** the solution, or the proof of non-existence of solutions, then we will care. `IP = PSPACE` is one such magical case where if someone can solve with a lot more power than you have access to, you can learn things from them interactively in reasonable time.

Chapter 3

Hierarchy Theorems

$\exists L$, such that $\forall f : \mathbb{N} \to \mathbb{N}$, where f is space/time constructible,

$$Space(f) \supsetneq Space(o(f))$$
$$Time(f) \supsetneq Time\left(\frac{o(f)}{log f}\right)$$

So, there is a Hierarchy of complexity classes in time and space.

3.0.1 Proof sketch

We exhibit a languag A, such that $A \in Space(f(n))$, and $A \notin Space(o(f(n))$.

Let D decide A. D's definition:

- compute $f(n)$ and mark the end of $f(n)$ cells. If the read-write head ever crosses it, `REJECT`, `HALT`. We first need $f(n)$ to use $f(n)$ cells or less to compute. This is called as **space-constructribility**. ($f : \mathbb{N} \to \mathbb{N}$ is space-constructible iff given n, $\exists$ TM which computes $f(n)$ using at most $f(n)$ cells). Also, we want $f(n)$ to be at least $log(n)$. Clearly, this process is in space $f(n)$.
- We now need to "seaparate" A from the smaller classes. If A can be solved in a smaller space (ie, we cannot separate A), then there must be a TM (say, D') which decides A in space less than $f(n)$. So now, we need to choose some input such that D' is different from D. We can use diagonalization to construct such a function.
- let the input be x. Let $x = M10*$ for some TM M. if not, `REJECT`, `HALT`.
- Let D simulate M on input M. If M takes less that $f(n)$ time to run on M, then M can decide A in time less than $f(n)$. So now, D knows how much space $M(\langle M \rangle)$ requires. if $M(\langle M \rangle)$ accepts, we reject. If $M(\langle M \rangle)$ rejects, we accept (diagonalization).
- To find out whether $M(\langle M \rangle)$ rejects, note that it is space-bounded, so we can just check how many states of the configuration space it visits. If it has not halted after visiting all states in the configuration space, we can conclude that $M(\langle M \rangle)$ does not halt. The configuration space is $O(2^{f(n)})$. So we need to run D for time $O(2^f(n))$, and then `REJECT` if it continues running.

Arjun Q: Are there examples of non-space constructible functions, which are non-trivial? Other than ones that are too-small?

Proofs of time are similar to the space separation theorem.

- compute $t(n)$ ($t(n)$ should be time-constructible). decrement a counter initialized to $t(n)$. if this hits 0, `REJECT, HALT`. We get a *log* factor due to the slowdown of keeping time. (People are trying to speed this up).
- once again, repeat the same construction used for $SPACE$.

3.1 Savitch's Theorem: $\texttt{NSPACE}(f(n)) \subseteq \texttt{PSPACE}(f(n)^2)$

$NSPACE(f(n))$ – one branch of a NTM N decides L in space $O(f(n)$.

Configuration space is $O(\texttt{alphabet}^{f(n)}) = O(2^f(n))$ – otherwise, configurations are repeated.

Our branch depth is exponential in $f(n)$. So, we need to keep track of $O(2^{f(n)})$ data.

Given $\langle C_1 \in Config(N), C_2 \in Config(N), t \in \mathbb{N} \rangle$ if we can find whether C_1 goes to C_2 in t space, then we can solve our original problem.

This can be solved by recursion by asking if there exists a C_{mid}, such that $C_1 \to C_{mid}$ in $t/2$ steps, similarly $C_{mid} to C_2$ in $t/2$ steps.

3.2 Cook Levin theorem

L is `NP-complete`, if

- $L \in NP$
- $\forall L' \in NP$, there exists a Karp reduction from L' to L: $L' \leq_p L$ (`NP-hard`)

$A \leq_p B$ if there exists a poly time computable function $f : \Sigma^* \to \Sigma^*$ such that

$$w \in A \Leftrightarrow f(a) \in B$$

Karp reduction = poly time mapping reduction.

Define `SAT`, and show that `SAT` is `NP-complete`.

We have boolean formulas ϕ, which is given in CNF.

$$\texttt{SAT} = \{\phi \mid \phi \text{ is in CNF (product of sums)}, \phi \text{ is satisfiable}\}$$

.

This is clearly decidable since we can try all possible assignments.

It is in NP since a NDTM can try to guess assignments.

To show that this is NP-complete, take any language L' in `NP`. We provide a karp reduction to `SAT`. We take the poly-time checker for L' into a SAT problem ψ, such that **iff** a solution for ψ exists, then the poly time checker will accept the string, and vice versa (for reject).

3.3 EXPSPACE completeness - $EQ_{REG\uparrow}$

$r \uparrow \equiv \exists k \in \mathbb{N}. r \uparrow$ is regular if r is regular. We need this operator to control input size.

Question: Check if two regular expressions are the same – We show that this $\notin$ PSPACE, and hence $\notin$ PTIME. We show that this problem is EXPSPACE complete.

Chapter 4

NP

4.1 Cook Levin theorem

`SAT` is `NP-hard`.

4.1.1 Proof

. Unfold theorem statement into: $\forall L \in \texttt{NP}, L \leq_p \texttt{SAT}$. Since this should work for all things in NP, let's just write down the definition:

there exists an `NDTM` N such that N accepts w, $\forall w \in L$, in $|w|^k$ steps.

N is an `NDTM`, so N accepts w means that there exists a branch of N that acceeps w is $|w|^k$ steps.

We should be able to construct a CNF such that $\phi(w)$ is `SAT` iff there exists an accepting branch for $N(w)$.

Caveats

1. the construction of ϕ from N should make sure that ϕ has $poly(|w|)$ clauses — otherwise, this is no longer a poly-time reduction. We know that $\langle \texttt{AND}, \texttt{OR}, \texttt{NOT} \rangle$ is universal, so we can clearly construct any `TM` into a circuit. The problem is that the CNF we construct from the truth-table of the `TM` will be polynomial.

Sid Q: Proof that boolean circuits are universal?

Proof sketch

Consider the NDTM $N(n)$, we will now argue about its configuration.

We can cut off the turing tape after the first polynomial number of cells — since the `NDTM` can only access those many cells.

We should start will the initial state q_{start}.

We should get the accept state q_{accept} in n^k steps.

If we can pose this in terms of a CNF formula of poly-length, we are done.

Setting up SAT

Variables - Cells of the tape The state of the turing tape on the ith step at the jth position of the turing tape for all $s \in alphabet(N)$ as $x_{i,j,s}$. $x_{i,j,s} = 1$ is interpreted as "at step i, on cell j, value s is written.

For this to be valid, we need each cell to have exactly one symbol.

Formula - Validity of cells For every (i,j) for at **least one** s must be 1: $\phi_{cellleast} = \bigwedge_{i,j}(\vee_s x_{i,j,s})$

For every (i,j) for at **most one** s must be 1. This is equivalent to saying that for every (s,t), one of them must be absent. $\phi_{cellmost} = \bigwedge_{s,t,s \neq t}(\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}})$.

$\phi_{cell} = \phi_{cellleast} \wedge \phi_{cellmost}$

Formula - Initial state $\phi_{init} = (x_{1,1,w_1} \wedge x1, 2, w_2 \wedge \cdots \wedge x_{1,n,w_n}) \wedge (x_{1,n+1,blank} \wedge x_{1,n+2,blank} \wedge \cdots \wedge x_{1,n^k,blank}$