15-455: UCT **K. Sutner**

Assignment 3 Due: **February 10**, 2024.

1. Kleene Star (25)

Background

For any language $L \subseteq \Sigma^\star$, recall the definition of the Kleene star of L:

$$L^\star = \{\, x_1 x_2 \ldots x_k \mid k \geq 0, x_i \in L \,\}$$

Define the marked Kleene star as

$$L^\star_\# = \{\, x_1 \# x_2 \# \ldots \# x_k \mid k \geq 0, x_i \in L \,\}$$

where $\#$ is a new symbol not in Σ.

It is well-known that $L^\star_\#$ and $L^\star$ are regular whenever L is regular and there is a simple algorithm to construct the corresponding finite state machines. Here is an analogous result for L being polynomial time.

Task

Assume that $L \in \mathbb{P}$.

A. Show that $L^\star_\#$ is in $\mathbb{P}$.

B. Show that $L^\star$ is in $\mathbb{P}$.

Comment

Don't try to argue directly in terms of Turing machines for this; just give a reasonable recognition algorithm for the two languages (and rely on our claim that implementation on a TM would only cause a polynomial slow-down).

2. Linear Time Reductions (20)

Background

A is linear time reducible to B if there is a linear time computable function $f : \Sigma^\star \to \Sigma^\star$ such that $x \in A \iff f(x) \in B$. One can show that this is a pre-order, so we can consider the corresponding equivalence relation $\equiv_{\mathsf{lin}}$, producing the linear time degrees.

Notation: $A \leq_{\mathsf{lin}} B$ and $A \equiv_{\mathsf{lin}} B$.

Recall the standard decision problems from graph theory:

Problem:	**Vertex Cover** (VC)
Instance:	A ugraph G, a bound k.
Question:	Does G have a vertex cover of cardinality k?

Problem:	**Independent Set** (IS)
Instance:	A ugraph G, a bound k.
Question:	Does G have an independent set of cardinality k?

Problem:	**Clique** (CL)
Instance:	A ugraph G, a bound k.
Question:	Does G have a clique of cardinality k?

Assume that the graph is given as an $n \times n$ Boolean matrix, k is written in binary. So the size of an instance (G, k) is $n^2 + \log n = \Theta(n^2)$ where n is the number of vertices in G.

Task

A. Show that linear time reducibility is a pre-order.

B. Show that $A \leq_{\mathsf{lin}} B$, $B \in \text{TIME}(n^k)$ implies $A \in \text{TIME}(n^k)$ for $k \geq 1$.

C. Show that $\mathsf{VC} \equiv_{\mathsf{lin}} \mathsf{IS} \equiv_{\mathsf{lin}} \mathsf{CL}$.

D. What would happen if we used an adjacency list representation instead? Say, the graph is written in the form

$$\#z\#x_1\#y_1\#x_2\#y_2\#\ldots\#x_e\#y_e\#$$

where e is the number of edges and the z, x_i, y_i are all k-bit numbers, $k = \max\big(1, \lceil \log(n+1) \rceil\big)$ and z is the binary expansion of n.

3. Collapsing Time (25)

Background

The Hartmanis/Stearns time hierarchy theorem says, in essence, that if g is smaller than f, then TIME(g) is properly contained in TIME(f). Here the running time function is required to be time constructible, so $f(n) \geq n$ (we generally require our machines to read the whole input).

Here is what happens when we drop that entirely reasonable condition.

Task

A. Find a natural arithmetical algorithm that, on input n, runs approximately in time $\widetilde{O}(\sqrt{n})$.

B. Show that TIME($\sqrt{n}$) = TIME(1).

C. Can you push the last result a bit?

Comment

For part (A), "natural" means that one could find the algorithm in number theory. Also note the "soft-Oh": you can ignore log factors. Argue in terms of actual algorithms, don't bother with Turing machines. But for part (B) you need to cope with the definition of TIME, and thus with Turing machines—but don't get bogged down.

4. Easy Satisfiability (30)

Background

Satisfiability is hard even if the given formula is in CNF and has exactly three literals per clause (3-CNF SAT). Still, there are several variants of SAT that are easy in the sense that they admit polynomial time algorithms.

The first one is easy to describe: in 2-CNF SAT one only has clauses containing exactly 2 literals. For the second, HornSAT, we are given a formula in CNF and each clause has at most one unnegated variable. The third variant, XorSAT, is also based on a kind of CNF, but this time the clauses are required to be exclusive ors of literals (ordinary disjunctions are not allowed).

So typical clauses in the three cases look like so:

$x_1 \vee \neg x_2$

$x_1 \vee \neg x_2 \vee \neg x_3$, $\neg x_1 \vee \neg x_2 \vee \neg x_3$

$x_1 \oplus \neg x_2 \oplus x_3$

Task

A. Show that 2-CNF SAT can be solved in polynomial time.

B. Show that HornSAT can be solved in polynomial time.

C. Show that XorSAT can be solved in polynomial time.