15-251: Great Theoretical Ideas In Computer Science

Recitation 3 Solutions

Announcements

- Office hours and Piazza are valuable resources. If you aren't sure what a question is asking, ask on Piazza or come to OH. Don't wait until the writing session.
- At the end of the writing session, you can hand in your homework in the unsorted box in the front of the room.

Regular or Not? You Decide

Define **REGULAR** $= \{\langle M \rangle \mid$ the set of strings accepted by M is a regular language$\}$.

Show that **REGULAR** is undecidable.

Assume that **REG** is decidable, by a decider R. We will construct a decider for the halting problem using R. Consider the following TM.

```
HALT(<M>,<w>) {
   HELP($x$) {
      run M(w)
      if $x \in \{0^n 1^n : n \in \mathbb{N}\}$, accept
      else, reject
   }
return NOT(R(<HELP>))
}
```

First, suppose that $M(w)$ halts. Then `HELP` acts as a decider for the language $\{0^n 1^n : n \in \mathbb{N}\}$ which is known to be irregular. So `R(<HELP>)` returns false and `HALT` returns true. Next, suppose that $M(w)$ does not halt. Then `HELP` accepts no strings, and $\emptyset$ is regular. So `R(<HELP>)` returns true and `HALT` returns false. Thus `HALT` is a decider for the halting problem which is a contradiction.

Counting sheep

For each set below, determine if it is countable or not. Prove your answers.

(a) $S = \{a_1 a_2 a_3 \ldots \in \{0,1\}^\infty \mid \forall n \geq 1$ the string $a_1 \ldots a_n$ contains more 1's than 0's.$\}$.

We offer two solutions to the student, one by diagonalization, and one by constructing an explicit injection from $\{0,1\}^\infty \to S$.

For diagonalization, suppose for the sake of contradiction that S were countable. Then there is some listing of the elements of S. We denote this list by the function $f : \mathbb{N} \to S$, so the ith element is just $f(i)$. We now construct an element $s \in S$ such that no input n maps $f(n) = s$. First, $s_1 = 1$, as must be true for any s. Now, for $i \in \mathbb{N}$, set $s_{2i+2} = 1$ and $s_{2i+3} = 1 - f(i)_{2i+3}$. First, we note that certainly there is no n such that $f(n) = s$, as $f(n)$ and s differ on the $2n+3$ digit, by construction. Furthermore, $s \in S$, as every even index is 1, and the first index is 1, so any prefix is guaranteed to have all even indices 1 (at least half), and one more to complete the strict inequality. As we have an element of S which is not in our list, we have a contradiction, so S must be uncountable.

For our second proof, we note that if we can construct $f : \{0,1\}^\infty \to S$ such that f is injective, then $|\{0,1\}^\infty| \leq |S|$, so as the former is uncountable, so is S. Let $f(s)$ be defined as follows:
Write $s = s_1 s_2 \cdots$.
$f(s) = 11s_1 1s_2 1s_3 1 \cdots$.
First, note this is injective, as taking every odd index but the first and concatenating together gives us a unique infinite binary string from the domain. Secondly, this result is in S, by construction, as every finite prefix of $f(s)$ has strictly more 1s than 0s, as every even index is a 1 (half the characters), with an extra 1 at the beginning to maintain strict inequality. As we have a valid injection, S is uncountable.

(b) Σ^*, where Σ is an alphabet that is allowed to be countably infinite (e.g., $\Sigma = \mathbb{N}$).

We apply the CS method here as well, representing each element as a finite string from a finite alphabet. Our alphabet will be the set of digits, as well as the comma character. To write down a finite string from Σ, first note that Σ is countable, and thus has an injection to $\mathbb{N}$. We can thus represent each string, instead of a concatenation of characters from sigma, as a comma-delimited list of natural numbers, each of which are representable as a finite string of digits. As we have a finite string from a finite alphabet representation, this set is countable.

Turing's Revenge

Determine whether the following languages are decidable or not. You may "use the Church–Turing Thesis" when proving your answers.

(a) $T = \{\langle M \rangle \mid \text{Turing machine } M \text{ accepts finitely many strings}\}$.

Suppose for the sake of contradiction we have some decider D for the set T. We proceed to solve the halting problem:

```
HALTS(<<M>,x>):
   HELP(<M'>):
      M(x)
      ACCEPT
   if D(<HELP>):
      REJECT
   else:
      ACCEPT
```

To show this works, first assume that $M(x)$ halts. Then, $HELP$ run on any input will ignore the input and accept. This means that $HELP$ accepts infinitely many strings, so $HELP \notin T$, so $D(< HELP >)$ rejects. By definition of our program, $HALTS(<< M >, x >)$ thus accepts, as desired.

Now, assume that $M(x)$ loops forever. Then, $HELP$ run on any input will ignore the input, then loop forever by running $M(x)$. This means that $HELP$ accepts the empty set of strings, which is finite. This implies that $D(< HELP >)$ accepts. By definition of our program, $HALTS(<< M >, x >)$ rejects, as desired.

In all cases, we return the correct thing, so we have decided the halting problem. This is impossible, as halting is undecidable, a contradiction, so this decider D cannot exist, so T is undecidable.

(b) $U = \{(\langle M \rangle, w) \mid M$ visits more than 251 distinct cells on its tape when processing $w\}$.

We will show that V is decidable.
Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$. Using the notation from problem 1, we can represent the configuration of the tape by uqv, where $q \in Q$, $u, v \in \Gamma*$.
If the TM uses fewer than 251 distinct tape cells, it will not be able to read an input longer than 251 characters. We can shorten the input, w to w', which is the first 251 characters of w. Our initial configuration is q_0w'.
If the TM uses fewer than 251 distinct tape cells, then in the computation history of $M(x)$, none of the configurations can have length greater than 502. In other words, $|u| + |v| \leq 502$.* Since both $|Q|$ and $|\Gamma|$ are finite, the total number of configurations where $|u| + |v| < 502$ is also finite. (It's on the order of $|\Gamma|^{503}|Q|$). Let C be the exact total number of configurations (a Turing machine can compute C by looking at each of the above configurations).
We can write the following decider, D.

```
D(<M>, w') :
    v =  # visited configurations
    s = $q_0w'$ # initial configuration
    while $|v| \leq C$:
        if s' is a halting configuration:
            REJECT
        s' = step(M, s)
        if s' in v: # loop detected
            REJECT
        if range(v) $\geq$ 251:
            ACCEPT
        insert s' in v
        s = s'
    ACCEPT # $>$ C distinct configurations visited
```

range(v) looks at all the configurations stored in v and returns the maximal distance between the position of the heads of two configurations.
We claim D decides V. First, D is a decider because it either rejects or accepts in finite time.
Suppose $M(w)$ visits no more than 251 tape cells. Then the distance between the heads of any two configurations can be at most 250. Also, $M(w)$ cannot visit more than C unique configurations, because there are only C unique ones. Thus, either $M(w)$ terminates before reaching C steps, or $M(w)$ loops. If $M(w)$ loops, then it will visit the same configuration twice, and we reject in that case too.
If $M(w)$ visits more than 251 tape cells, then the maximal distance between the heads of two configurations must be at least 251, and we accept in this case. Similarly, if we are able to visit more than C unique configurations, then we were able to visit more than 251 distinct tape cells, and we accept in this case too.