These Decidable Definitions Have Undecidable Ends

- A **decider** is a TM that halts on all inputs.
- A language \( L \) is **undecidable** if there is no TM \( M \) that halts on all inputs such that \( M(x) \) accepts if and only if \( x \in L \).
- A language \( A \) reduces to \( B \) if it is possible to decide \( A \) using an algorithm that decides \( B \) as a subroutine. Denote this as \( A \leq B \) (read: \( B \) can be used to solve \( A \) so \( A \) is at most as hard as \( B \)).
- Countability cheat sheet: You are given a set \( A \). Is it countable or uncountable?

\[
egin{align*}
|A| &\leq |N| \quad (A \text{ is countable}) \\
&- \text{Show directly an injection from } A \text{ to } N \quad (A \hookrightarrow N) \text{ or a surjection from } N \text{ onto } A \\
&\quad (N \twoheadrightarrow A) \\
&- \text{Show } |A| \leq |B|, \text{ where } B \text{ is one of } Z, Z \times Z, Q, \Sigma^*, Q[x], \text{ etc.}
\end{align*}
\]

*This one is important and very powerful*

\[
egin{align*}
|A| &> |N| \quad (A \text{ is uncountable}) \\
&- \text{Show directly using a diagonalization argument.} \\
&- \text{Show that } |\{0,1\}^\infty| \leq |A|, \text{ i.e. an injection from } \{0,1\}^\infty \text{ to } A.
\end{align*}
\]

Counting sheep

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

(a) \( S = \{a_1a_2a_3 \ldots \in \{0,1\}^\infty \mid \forall n \geq 1 \text{ the string } a_1 \ldots a_n \text{ contains more 1's than 0's.} \} \)

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

For our first 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 \( s \in \{0,1\}^\infty \) and \( i \in \mathbb{N} \). Write \( s = s_1s_2 \cdots \), and define \( f(s)_i \) to be

\[
f(s)_i = \begin{cases} 
1 & \text{if } i = 1 \text{ or } i \text{ is even} \\
\frac{s_{(i-1)/2}}{2} & \text{otherwise}
\end{cases}
\]
Informally, \( f(s) = 11s_1s_21\cdots \). First, we show that this is injective. Suppose \( s, s' \in \{0, 1\}^{\infty} \) such that \( s \neq s' \). Then if we write \( s = s_1s_2\cdots \) and \( s' = s'_1s'_2\cdots \), then there must be some index \( i \) at which \( s_i \neq s'_i \). Then \( f(s)_{2i+1} = s_i \neq s'_i = f(s')_{2i+1} \), and so \( f(s) \neq f(s') \). So \( f \) is injective.

Secondly, we show that \( f(s) \) is in \( S \). Let \( s \in \{0, 1\}^{\infty} \) and \( n \geq 1 \). Clearly, the condition holds for \( n = 1 \) and \( n = 2 \). Now consider \( n \geq 3 \). If \( n \) is even, then there will be at least \( \frac{n}{2} + 1 \) 1's by construction, so there are definitely more 1's than 0's in the first \( n \) digits. If \( n \) is odd, then there will be at least \( \frac{n+1}{2} = \frac{n}{2} + \frac{1}{2} \) 1's by construction, so there still are definitely more 1's than 0's in the first \( n \) digits. Thus \( f(s) \in S \). As we have a valid injection, \( S \) is uncountable.

For diagonalization, suppose for the sake of contradiction that \( S \) were countable. In particular, this means that there is a bijection \( f : \mathbb{N} \to S \). We now construct an element \( s \in S \) such that no input \( n \) is mapped to \( s \).

Define \( s \) as follows:

\[
s_i = \begin{cases} 
1 & \text{if } i = 1 \text{ or } i \text{ even} \\
1 - f \left( \frac{i-1}{2} \right) & \text{otherwise}
\end{cases}
\]

First, we note that for any \( n \), \( f(n) \) differs from \( s \) at the \( 2n + 1 \) digit, by construction. Furthermore, \( s \in S \) for the same reason as the construction of \( f \) in the previous section. As we have an element of \( S \) which is not in our list, we have a contradiction, so \( S \) must be uncountable.

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

We apply the CS method (countability = encodability), 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 \( \Sigma \), first note that \( \Sigma \) is countable, and so there is an injection from \( \Sigma \) 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.

**Doesn’t Look Like Anything (Decidable) To Me**

Prove that the following languages are undecidable (below, \( M, M_1, M_2 \) refer to TMs).

(a) \( \text{REGULAR} = \{ \langle M \rangle : L(M) \text{ is regular} \} \).

We show that \( \text{REGULAR} \) is undecidable via a reduction from \( \text{HALTS} \). Suppose \( M_{\text{REG}} \) decides \( \text{REGULAR} \). We define a decider for \( \text{HALTS} \) as follows

```python
def M_HALTS(<M, x>):
    <HELP> =
    """def HELP(w):
        if w in {0^n1^n | n a natural number}:
            ACCEPT
        else:
            M(x)
    ACCEPT"
```

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

We apply the CS method (countability = encodability), 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 \( \Sigma \), first note that \( \Sigma \) is countable, and so there is an injection from \( \Sigma \) 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.
return M_REG(⟨HELP⟩)

Proof of correctness:
Suppose that $M(x)$ halts, then $L(HELP) = \Sigma^*$, so $M_REG(⟨HELP⟩)$ accepts, as desired.
Suppose that $M(x)$ loops, then $L(HELP) = \{0^n1^n | n \in \mathbb{N}\}$, so $M_REG(⟨HELP⟩)$ rejects, as desired.
Thus, we’ve shown that $HALTS \leq REGULAR$, so $REGULAR$ is undecidable.

(b) $TOTAL = \{⟨M⟩| M$ halts on all inputs$\}$.

We show that $TOTAL$ is undecidable via a reduction from $HALTS$. Suppose $M_{TOTAL}$ decides $TOTAL$. We define a decider for $HALTS$ as follows

```python
def M_HALTS(<M, x>):
    <HELP> = 
    """def HELP(w):
    M(x)
    ACCEPT"
    return M_TOTAL(<HELP>)
```

Proof of correctness:
Suppose that $M(x)$ halts, then HELP halts on all inputs, so $M_{TOTAL}(⟨HELP⟩)$ accepts, as desired.
Suppose that $M(x)$ loops, then HELP does not halt on all inputs, so $M_{TOTAL}(⟨HELP⟩)$ rejects, as desired.
Thus, we’ve shown that $HALTS \leq TOTAL$, so $TOTAL$ is undecidable.

(c) $DOLORES = \{⟨M_1, M_2⟩ : \exists w \in \Sigma^* such that both M_1(w) and M_2(w) accept$\}.

We show that $DOLORES$ is undecidable via a reduction from $HALTS$. Suppose $M_{DOLORES}$ decides $DOLORES$. We define a decider for $HALTS$ as follows

```python
def M_HALTS(<M, x>):
    <HELP> = 
    """def HELP(w):
    M(x)
    ACCEPT"
    return M_DOLORES(<HELP, HELP>)
```

Proof of correctness:
Suppose that $M(x)$ halts, then HELP accepts all inputs, so $M_{DOLORES}(⟨HELP, HELP⟩)$ accepts, as desired.
Suppose that $M(x)$ loops, then HELP rejects all inputs, so $M.DOLORES(\langle HELP, HELP \rangle)$ rejects, as desired.

Thus, we’ve shown that $\text{HALTS} \leq \text{DOLORES}$, so $\text{DOLORES}$ is undecidable.

**(Extra) Lose All Scripted Responses. Improvisation Only**

Let $\text{FINITE} = \{\langle M \rangle : M \text{ is a TM and } L(M) \text{ is finite} \}$.

Show that $\text{TOTAL} \leq \text{FINITE}$.

Suppose that $M.FINITE$ decides $\text{FINITE}$.

We define a decider for $\text{TOTAL}$ as follows, where $<$ means lexicographically smaller.

```python
def M_TOTAL(<M>):
    <HELP> =
    """def HELP(w):
        for y <= w:
            M(y)
            ACCEPT"
    return not M_FINITE(<HELP>)
```

Proof of correctness:

Suppose $M$ is total and halts on all inputs, then $L(HELP) = \Sigma^*$, so $M.FINITE(\langle HELP \rangle)$ will reject, as desired.

Suppose $M$ is not total and let $w$ be the lexicographically smallest string that $M$ does not halt on. Then $HELP$ will not accept any string lexicographically greater (or equal to) than $w$, as $M(w)$ will not halt. Thus, since there are only finitely many strings lexicographically smaller than $w$, $L(HELP)$ is finite. Thus, $M.FINITE(\langle HELP \rangle)$ accepts, as desired.

**(Bonus) The Maize is not Meant For You**

Josh Corn is trying to write a program $P$ such that given a natural number $n$, $P(n)$ is the most number of steps a TM on $n$ states can take before halting. Show that this is not possible.