Important Definitions

The terms **recursive** and **recursively enumerable** apply both to sets of numbers and to sets of strings (i.e., languages).

A set $S$ is **recursive** if:

- There is a Turing Machine which recognizes $S$; or, equivalently,
- $S$ has a total recursive characteristic function

A set $S$ is **recursively enumerable** if:

- There is a Turing Machine which accepts $S$; or, equivalently,
- There is a Turing Machine which generates $S$

All of these definitions apply both to sets of numbers and sets of strings. Additionally, if $S$ is a set of numbers:

Set $S$ is **recursively enumerable** if

- $S$ is the domain of some partial recursive function $f$.

We can also extend some the decidable/undecidable terminology from Rice’s Theorem to problems in general:

A yes/no problem is **decidable** if there is a Turing Machine which always halts and correctly answers 1 ("yes") or 0 ("no") to the problem.

A yes/no problem is **undecidable** if there is no Turing Machine which always halts and correctly answers 1 ("yes") or 0 ("no") to the problem.

Because of the Church-Turing thesis, you can substitute "algorithm" for "Turing Machine" in each of the above definitions.

**Review Questions: Decidability**

Note: Some of these questions are very easy, and are listed just to see that you understand the definitions involved.

1. Classify each of the following problems as **decidable** or **undecidable**. Justify your answer in each case. (To show the existence of a Turing machine for this question, it is sufficient to describe an algorithm in English; you do not have to give the full state diagram.) $L(M)$ means "the language accepted by Turing Machine $M$".

   a) Given a Turing machine $M$, is $L(M)$ recursively enumerable?

   b) Given a Turing machine $M$, does $L(M)$ have a recursively enumerable complement?

   c) Given a Turing machine $M$ with input alphabet $\Sigma = \{a, b\}$, does $L(M)$ contain the empty string?

   d) Given a string $x$ (over alphabet $\Sigma = \{a, b\}$), is $x$ the empty string?

2. Classify each of the following languages as **recursive**, **recursively enumerable but not recursive**, or **not recursively enumerable**. Justify your answer in each case. (As before, to show the existence of a Turing machine for this question, it is sufficient to give an algorithm.)

   Assume that the alphabet for languages (a)-(e) is $\{1\}$, with each $x$ encoded in unary-plus-one notation; that the input alphabet for each Turing Machine mentioned in the definitions is $\{a, b\}$; and that the Gödel encodings are designed for this alphabet.

   a) $\{x \mid x \text{ is the Gödel number of some Turing Machine.}\}$

   b) $\{x \mid x \text{ is the Gödel number of a Turing Machine } M, \text{ and } L(M) \text{ is recursively enumerable}\}$

   c) $\{x \mid x \text{ is the Gödel number of a Turing Machine } M, \text{ and } L(M) \text{ contains the string } aaa\}$

   d) $\{x \mid x \text{ is the Gödel number of a Turing Machine } M, \text{ and } L(M) \text{ does not contain the string } aaa\}$

   e) $\{x \mid x \text{ is the Gödel number of a Turing Machine } M, \text{ and } L(M) \subseteq a^*\}$
Review Questions: Chomsky Hierarchy

Assume that the alphabet for each of the languages below is \{a, b, c\}.

3. Design a nondeterministic finite automaton that accepts \((abc)^*a(b^*)c\).
4. Design a right-linear grammar that accepts \((abc)^*a(b^*)c\).
5. Design a nondeterministic finite automaton that accepts \(a^*b^*c^*\).
6. Design a right-linear grammar that accepts \(a^*b^*c^*\).
7. Design a nondeterministic finite automaton that accepts \((abc)^*a(b^*)c \cup a^*b^*c^*\). Be as lazy as possible: reuse as much of your solutions from #3 and #5 as you can.
8. Design a right-linear grammar that accepts \((abc)^*a(b^*)c \cup a^*b^*c^*\). Be as lazy as possible: reuse as much of your solutions from #4 and #6 as you can.
9. Construct a context-free grammar that accepts the same language as this context-sensitive grammar:

\[
\begin{align*}
S & \rightarrow A \ B \\
A & \rightarrow a \\
A & \rightarrow c \\
a \ B & \rightarrow a \ B \ B \\
a \ B & \rightarrow a \ b \\
B & \rightarrow b 
\end{align*}
\]

10. Show that the set \(Odd\) of odd natural numbers is recursively enumerable by showing that \(Odd\) is \(Image(f)\) for some unary total recursive function \(f\).

11. 12.7.2 a, b, c, l

Hint for 12.7.2 a: Given a finite set of strings, can you construct a grammar that accepts it? What does the grammar look like? How about a regular expression or finite automaton: can you make one of these to accept the set?

Hint for 12.7.2 l: Consider one member of this family, the set of strings over \{a, b\} of length 4. How many strings are in this set?