

Generally useful information.

- When I say a set S is re, unless I explicitly say otherwise, you may assume any of the following equivalent characterizations:
 1. S is either empty or the range of a total recursive function f_S .
 2. S is the domain of a partial recursive function g_S .
- If I say a function g is partially computable, then there is an index g (I know that's overloading, but that's okay as long as we understand each other), such that $\Phi_g(x) = \Phi(x, g) = g(x)$. Here Φ is a universal partially recursive function.
 Moreover, there is a primitive recursive function STP , such that $STP(x, g, t)$ is 1 (true), just in case g , started on x , halts in t or fewer steps.
 $STP(x, g, t)$ is 0 (false), otherwise.
 Finally, there is another primitive recursive function $VALUE$, such that $VALUE(x, g, t)$ is $g(x)$, whenever $STP(x, g, t)$.
 $VALUE(x, g, t)$ is defined but meaningless if $\sim STP(x, g, t)$.
- The notation $f(x)\downarrow$ means that f converges when computing with input x , but we don't care about the value produced. In effect, this just means that x is in the domain of f .
- The notation $f(x)\uparrow$ means f diverges when computing with input x . In effect, this just means that x is **not** in the domain of f .
- The Post Correspondence problem is known to be undecidable. This problem is characterized by instances that are described by two n -ary sequences of non-empty words $\langle x_1, x_2, \dots, x_n \rangle, \langle y_1, y_2, \dots, y_n \rangle$. The question is whether or not there exists a sequence, i_1, i_2, \dots, i_k , such that $x_{i_1}x_{i_2}\dots x_{i_k} = y_{i_1}y_{i_2}\dots y_{i_k}$
- The language $\{ww \mid w \text{ is a word in some alphabet with more than one letter}\}$ is not a CFL.
 The language $\{a^n b^n c^n \mid n \geq 0\}$ is not a CFL. The language $\{a^n b^n \mid n \geq 0\}$ is not **Regular**.
- When I ask for a reduction of one set of indices to another, the formal rule is that you must produce a function that takes an index of one function and produces the index of another having whatever property you require. However, I allow some laxness here. You can start with a function, given its index, and produce another function, knowing it will have a computable index. For example, given f , a unary function, I might define g_f , another unary function, by $g_f(0) = f(0)$
 $g_f(y+1) = g_f(y) + f(y+1)$
 This would get $g_f(x)$ as the sum of the values of $f(0)+f(1)+\dots+f(x)$, which could be useful.

1. Choosing from among **(D) decidable**, **(U) undecidable**, **(?) unknown**, categorize each of the following decision problems. No proofs are required.

Problem / Language Class	Regular	Context Free	Context Sensitive
$L = \Sigma^*$?			
$L = \phi$?			
$L = L^2$?			
$x \in L^2$, for arbitrary x ?			

2. Choosing from among **(Y) yes**, **(N) No**, **(?) unknown**, categorize each of the following closure properties. No proofs are required.

Problem / Language Class	Regular	Context Free
Closed under intersection?		
Closed under quotient?		
Closed under quotient with Regular languages?		
Closed under complement?		

3. Consider the two operations on languages **max** and **min**, where
 $\max(L) = \{ x \mid x \in L \text{ and, for no } y \text{ does } xy \in L \}$ and
 $\min(L) = \{ x \mid x \in L \text{ and, for no prefix of } x, y, \text{ does } y \in L \}$
 Describe the languages produced by **max** and **min**. for each of the following:

$$L_1 = \{ a^i b^j c^k \mid k \leq i \text{ or } k \leq j \}$$

$$\max(L_1) = \underline{\hspace{10em}}$$

$$\min(L_1) = \underline{\hspace{10em}}$$

$$L_2 = \{ a^i b^j c^k \mid k > i \text{ or } k > j \}$$

$$\max(L_2) = \underline{\hspace{10em}}$$

$$\min(L_2) = \underline{\hspace{10em}}$$

4. Prove that any class of languages, C , closed under union, concatenation, intersection with regular languages, homomorphism and substitution (e.g., the Context-Free Languages) is closed under **MissingMiddle**, where, assuming L is over the alphabet Σ ,

$$\mathbf{MissingMiddle}(L) = \{ xz \mid \exists y \in \Sigma^* \text{ such that } xyz \in L \}$$

You must be very explicit, describing what is produced by each transformation you apply.

5. Use **PCP** to show the undecidability of the problem to determine if the intersection of two context free languages is non-empty. That is, show how to create two grammars G_A and G_B based on some instance $P = \langle \langle x_1, x_2, \dots, x_n \rangle, \langle y_1, y_2, \dots, y_n \rangle \rangle$ of **PCP**, such that $L(G_A) \cap L(G_B) \neq \emptyset$ iff P has a solution. Assume that P is over the alphabet Σ . You should discuss what languages your grammars produce and why this is relevant, but no formal proof is required.

6. Consider the set of indices $\mathbf{CONSTANT} = \{ f \mid \exists K \forall y [\varphi_f(y) = K] \}$. Use Rice's Theorem to show that $\mathbf{CONSTANT}$ is not recursive. Hint: There are two properties that must be demonstrated.

7. Show that $\mathbf{CONSTANT} \equiv_m \mathbf{TOT}$, where $\mathbf{TOT} = \{ f \mid \forall y \varphi_f(y) \downarrow \}$.

8. Why does Rice's Theorem have nothing to say about each of the following? Explain by showing some condition of Rice's Theorem that is not met by the stated property.

a.) **AT-LEAST-LINEAR** = { $f \mid \forall y \ \varphi_f(y)$ converges in no fewer than y steps }.

b.) **HAS-IMPOSTER** = { $f \mid \exists g \ [g \neq f \text{ and } \forall y \ [\varphi_g(y) = \varphi_f(y)]]$ }.

9. The trace language of a computational device like a Turing Machine is a language of the form

Trace = { $C_1\#C_2\# \dots C_n\# \mid C_i \Rightarrow C_{i+1}, 1 \leq i < n$ }

Trace is Context Sensitive, non-Context Free. Actually, a trace language typically has every other configuration word reversed, but the concept is the same. Oddly, the complement of such a trace is Context Free. Explain what makes its complement a **CFL**. In other words, describe the characteristics of this complement and why these characteristics are amenable to a **CFG** description.

Note: Reversing the second word in a pair is important here if you're thinking about Turing Machines but is irrelevant for **FRS with Residue**. Thus, don't make reversal an issue in your discussion. Also, here's a start for you, assuming Σ is the alphabet of configuration words.

Let $R = (\Sigma + \#)^*##(\Sigma + \#)^* + \Sigma^* + \#(\Sigma + \#)^* + (\Sigma + \#)^*\Sigma^+$

R is a regular expression that describes all words that do not look like sequences of configurations. Your job now is to describe **BadTrace**, the rest of the complement of **Trace** and discuss why it's a **CFL**.