**COT 6410 Spring 2014 Final Exam E1 Sample Name:**

* The notation **z =** **<x,y>** denotes the pairing function with inverses **x =** **<z>1** and **y =** **<z>2**.
* The minimization notation **μ y [P(…,y)]** means the least **y** (starting at **0**) such that **P(…,y)** is true. The bounded minimization (acceptable in primitive recursive functions) notation   
  **μ y (u≤y≤v) [P(…,y)]** means the least **y** (starting at **u** and ending at **v**) such that **P(…,y)** is true. I define **μ y (u≤y≤v) [P(…,y)]** to be **v+1**, when no **y** satisfies this bounded minimization.
* The tilde symbol, **~,** means the complement. Thus, set **~S** is the set complement of set **S**, and the predicate **~P(x)** is the logical complement of predicate **P(x).**
* A function **P** is a predicate if it is a logical function that returns either **1** (**true**) or **0** (**false**). Thus, **P(x)** means **P** evaluates to **true** on **x**, but we can also take advantage of the fact that **true** is **1** and **false** is **0** in formulas like **y × P(x)**, which would evaluate to either **y** (if **P(x)**) or **0** (if **~P(x)**).
* A set **S** is recursive if **S** has a total recursive characteristic function **χS**, such that **x ∈ S ⇔ χS(x)**. Note **χS** is a total predicate. Thus, it evaluates to **0** (**false**), if **x ∉ S**.
* 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 **fS**.
2. **S** is the domain of a partial recursive function **gS**.

* If I say a function **g** is partially computable, then there is an index **g** (we tend to overload the index as the function name), such that **Φg(x) = Φ(x, g) = g(x)**. Here **Φ** is a universal partially recursive function.   
  Moreover, there is a primitive recursive function **STP**, such that   
  **STP(g, x, t)** is **1** (true), just in case **g**, started on **x**, halts in **t** or fewer steps.   
  **STP(g, x, t)** is **0** (false), otherwise.   
  Finally, there is another primitive recursive function **VALUE**, such that   
  **VALUE(g, x, t)** is **g(x)**, whenever **STP(g, x, t)**.   
  **VALUE(g, x, t)** is defined but meaningless if **~STP(g, x, t)**.
* The notation **f(x)↓** means that **f** converges when computing with input **x** (**x ∈ Dom(f)**). The notation **f(x)↑** means **f** diverges when computing with input **x** (**x ∉ Dom(f)**).
* The **Halting Problem** for any effective computational system is the problem to determine of an arbitrary effective procedure **f** and input **x**, whether or not **f(x)↓**. The set of all such pairs, **K0**, is a classic re non-recursive set. **K0** is also known as **Lu**, the universal language. The related set, **K**, is the set of all effective procedures **f** such that **f(f)↓** or more precisely **Φf(f)**.
* The **Uniform Halting Problem** is the problem to determine of an arbitrary effective procedure **f**, whether or not **f** is an algorithm (halts on all input). This set, **TOTAL**, is a classic non re set.
* 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 **Gf**, another unary function, by   
  **Gf(0) = f(0); Gf(y+1) = Gf(y) + f(y+1)**This would get **Gf(x)** as the sum of the values of **f(0)+f(1)+…+f(x)**.
* The **Post Correspondence Problem** (**PCP**) is known to be undecidable. This problem is characterized by instances that are described by a number **n>0** and two **n**-ary sequences of non-empty words **<x1,x2,…,xn>, <y1,y2,…,yn>**. The question is whether or not there exists a sequence, **i1,i2,…,ik**, such that **1≤ij≤n**, **1≤j≤k**, and **xi1xi2…xik = yi1yi2…yik**
* When I ask you to show one set of indices, **A**, is many-one reducible to another, **B**, denoted   
  **A ≤m B,** you must demonstrate a total computable function **f**, such that **x ∈ A ⇔ f(x) ∈ B**. The stronger relationship is that **A** and **B** are many-one equivalent, **A ≡m B**, requires that you show   
  **A ≤m B** and **B ≤m A**. The related notion of one-one reducibility and equivalence require that the reducing function, **f** above, be 1-1. The notation just replaces the **m** with a **1**, as in **A ≤1 B**.

**COT 6410 Spring2014 Final Exam Sample E1 Questions**

**1**. Let set **A** be recursive, **B** be re non-recursive and **C** be non-re. Choosing from among **(REC)** **recursive**, **(RE)** **re non-recursive**, **(NR)** **non-re**, categorize the set **D** in each of a) through d) by listing **all** possible categories. No justification is required.

**a.) D = ~C**

**b.) D ⊆ (A∪C)**

**c.) D = ~B**

**d.) D = B − A**

**2**. 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 = Σ\* ? |  |  |  |
| **L = φ ?** |  |  |  |
| **L = L2 ?** |  |  |  |
| **x ∈ L2, for arbitrary x ?** |  |  |  |

**3.** 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 **GA** and **GB** based on some instance **P = <<x1,x2,…,xn>, <y1,y2,…,yn>>** of **PCP**, such that **L(GA) ∩ L(GB) ≠ φ** 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.

**4.** Consider the set of indices **CONSTANT = { f | ∃K ∀y [ ϕf(y) = K ] }**. Use Rice’s Theorem to show that **CONSTANT** is not recursive. Hint: There are two properties that must be demonstrated.

**5.** Show that **CONSTANT ≡m TOT**, where **TOT = { f | ∀y ϕf(y)↓ }**.

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

**AT-LEAST-LINEAR = { f | ∀y ϕf(y) converges in no fewer than y steps }**.

**7.** The trace language of a computational device like a Turing Machine is a language of the form  
**Trace = { C1#C2# … Cn# | Ci ⇒ Ci+1, 1 ≤ 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.

**8.** We demonstrated a proof that the context sensitive languages are not closed under homomorphism, To start, we assumed **G = (N, Σ, S, P)** is an arbitrary Phrase Structured Grammar, with **N** its set of non-terminals, **Σ** its terminal alphabet, **S** its starting non-terminal and **P** its productions (rules). Since **G** is a PSG, it can have length increasing, length preserving and length decreasing rules. We wished to convert **G** to a CSG, **G’ = (N’, Σ’, S’, P’)** where there are no rules that are length decreasing (since a CSG cannot have these). We developed a way to pad the length decreasing rules from **G** and then a homomorphism that gets rid of these padding characters. Define **G’** and the homomorphism **h** that we discussed in class and then briefly discuss why this new grammar and homomorphism combine so **h(L(G’)) = L(G)**, thereby showing that all re sets are the homomorphic images of CSLs.