COT 6410 Spring 2020 Legal Cheat Sheet Name:__________________

- Formal languages are categorized via a hierarchy from the simplest to the more complex – regular, content-free, context sensitive, phrase structured. Each class in the hierarchy is properly contained in the next with the phrase-structured being equivalent to class of re (Turing recognizable) languages.
- Every class of formal languages has an associated accepting automata family and an associated generative grammar family.
- The Regular Languages, in addition to being recognized by finite state automata and generated by either right or left linear grammars, can be specified by regular expressions, the solutions to regular equations and the union of some of the classes of a right-invariant equivalence relation. Moreover, Regular Languages have unique minimal state deterministic recognizers.
- Remember that Context Free Languages can be generated by Context Free Grammars and recognized by non-deterministic Pushdown Automata. Also, membership of a string **w** in the language associated with a CFG can be checked in ($\mathbf{n^3}$), where $\mathbf{n = |w|}$. (Really in $\mathbf{n^{2.37}}$).
- Remember that Context Sensitive Grammar rules are non-length reducing, but Phrase Structured Grammars may have length-reducing rules. Also, recall that Context Sensitive Languages are generated by Context Sensitive Grammars and recognized by Linear Bounded Automata. Phrase Structured Languages are generated by Phrase Structured Grammars and recognized by Turing Machines.
- The languages **{ww | w** is a word in some alphabet with more than one letter}, $\mathbf{\{a^nb^nc^n \mid n\geq 0\}}$, $\mathbf{\{a^ib^jc^k \mid k = \min(i,j)\}}$, and $\mathbf{\{a^ib^jc^k \mid k = \max(i,j)\}}$ are not **CFLs**. All of these are, however, **CSL**s. The languages $\mathbf{\{ww^R \mid w}$ is a word in some alphabet with more than one letter}, $\mathbf{\{xy \mid |x| = |y|, x \neq y,\ x,y}$ are words in some alphabet with more than one letter**}** **a**nd $\mathbf{\{a^nb^n \mid n\geq 0\}}$ are **CFLs**, but none is **Regular**.
- Languages can be shown non-regular using the Pumping Lemma for Regular Languages or by employing Myhill-Nerode. Languages can be shown non-context-free using the Pumping Lemma for Context-Free Languages.
- Non-determinism is necessary for PDAs to recognize all CFLs, but adds nothing to the power of finite state automata, linear-bounded automata or Turing Machines. Oddly non-determinism detracts from the power of Factor Replacement System.
- The notation $\mathbf{z = <x,y>}$ denotes some 1-1 onto pairing function with inverses $\mathbf{x = <z>_1}$ and $\mathbf{y = <z>_2}$.
- The notation $\mathbf{f(x)\downarrow}$ 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 $\mathbf{f(x)\uparrow}$ means **f** diverges when computing with input **x**. In effect, this just means that **x** is **<u>not</u>** in the domain of **f**.
- The minimization notation $\mathbf{\mu\, y\, [P(\ldots,y)]}$ means the least **y** (starting at **0**) such that **P(…,y)** is true. The bounded minimization (acceptable in primitive recursive functions) notation $\mathbf{\mu\, y\, (u\leq y\leq v)\, [P(\ldots,y)]}$ means the least **y** (starting at **u** and ending at **v**) such that **P(…,y)** is true. I define $\mathbf{\mu\, y\, (u\leq y\leq v)\, [P(\ldots,y)]}$ to be **v+1**, when no **y** satisfies this bounded minimization.
- 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 $\mathbf{y \times P(x)}$, which would evaluate to either **y** (if **P(x)**) or **0** (if **not P(x)**).
- 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 set **S** is recursive (decidable) if **S** has a total recursive characteristic function χ_S, such that $\mathbf{x \in S \Leftrightarrow \chi_S(x)}$. Note χ_S is a total predicate. Thus, it evaluates to **0** (**false**), iff $\mathbf{x \notin S}$.

- 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 (algorithm) $\mathbf{f_S}$.
  2. **S** is the domain of a partial recursive function (one that may diverge on some input) $\mathbf{g_S}$.
  3. **S** is the range of a partial recursive function (one that may diverge on some input) $\mathbf{h_S}$.
  4. **S** is recognizable by a Turing Machine.
  5. **S** is the language generated by a phrase structured grammar.
- 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 $\varphi_g(x) = \varphi(g, x) = g(x)$. Here φ is a universal partial recursive function (an interpreter).
  Moreover, there is a primitive recursive predicate **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 **Halting Problem** for any effective computational system is the problem to determine of an arbitrary effective procedure **f** and input **x**, whether or not $f(x)\downarrow$. The set of all such pairs, K_0, is a classic re non-recursive set. K_0 is also known as L_u, the universal language. The related set, **K**, is the set of all effective procedures **f** such that $f(f)\downarrow$ or more precisely $\Phi_f(f)$. **K** and K_0 are classic **RE-Complete** sets, meaning that every **RE** set many-one reduces these hardest **RE** sets.
- The **Uniform Halting Problem** is the problem to determine of an arbitrary effective procedure **f**, whether **f** is an algorithm (halts on all input). The set of all such function indices is a classic non-**re** one and is often called **TOTAL**. It can be described as $\{ f \mid \forall x\ \varphi_f(x)\downarrow \}$.
- When I ask you to show one set of indices, **A**, is many-one reducible (or simply reducible) to another, **B**, denoted $A \le_m B$ (or simply $A \le B$), you must demonstrate a total computable function **f**, such that $x \in A \Leftrightarrow f(x) \in B$. The stronger relationship that **A** and **B** are many-one equivalent, $A \equiv_m B$, requires that you show $A \le_m B$ and $B \le_m A$.
- In the computability domain, we usually categorize problems as **Recursive**, **Recursively Enumerable (RE)**, **Co-Recursively Enumerable (co-RE)**, **RE-Complete**, **Co-RE-Complete**, and **NRNC** (my own term to describe a set for which we can show a reduction from some **RE-Complete**, e.g., **TOTAL** is in **NRNC** since $K \le_m$ **TOTAL**).
- While reduction is a general tool to show undecidability, Rice's Theorem, though constrained, is a very easy tool to employ, provided you use it properly. Quantification is also great for estimating complexity; well at least to put an upper bound.
- The **Post Correspondence Problem** (**PCP**) is known to be undecidable. This problem is characterized by instances that are described by a finite alphabet, Σ, a number $n>0$ and two **n**-ary sequences of non-empty words $<x_1,x_2,\dots,x_n>$, $<y_1,y_2,\dots,y_n>$, each in Σ^+. The question is whether or not there exists a sequence, $i_1,i_2,\dots,i_k$, such that $1\le i_j\le n$, $1\le j\le k$, and $x_{i_1}x_{i_2}\cdots x_{i_k} = y_{i_1}y_{i_2}\cdots y_{i_k}$
- **PCP** can be used to show the undecidability of Ambiguity in CFGs, the undecidability of two CFGs producing overlapping words and the undecidability of non-emptiness problem for CSGs.
- The undecidability of a CFG producing Σ^* is based on traces of terminating computations in Turing machines. Really, it's based on the complements of such traces. Traces can also show the complexity of the quotient of CFLs and of power problems for CFLs.

COT 6410 Spring 2020 **Midterm#1** **Name:__________________**

Raw Score ___ / 60 **Grade:_________**

6 **1**. In each case below, consider **R1** to be Regular, **R2** to be finite, and **L1** and **L2** to be non-regular CFLs. Fill in the three columns with **Y** or **N**, indicating what kind of language **L** can be. No proofs are required. Read ⊆ as "contained in and may equal."
Put **Y** in all that are possible and **N** in all that are not.

Definition of L	**Regular?**	**CFL, non-Regular?**	**Not even a CFL?**
L = L1 / L2			
L = L1 – R1			
L = Σ* – L1			
L ⊆ R2			

3 **2**. Choosing from among **(D) decidable**, **(U) undecidable**, **(?) unknown**, categorize each of the following decision problems. No proofs are required. **L** is a language over **Σ**; **w** is a word in **Σ***

Problem / Language Class	**Regular**	**Context Free**	**Context Sensitive**	**Phrase Structured**
L = ∅ ?				
L is Σ* ?				

4 **3**. 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 **Double Interior Retention with Regular Sets**, denoted by the operator ||, where **L** ∈ *C*, **R** is Regular, **L** and **R** are both over the alphabet **Σ**, and
L||R = { vx | v,x ∈ R and ∃u,w ∈ Σ^+ such that uvwx ∈ L }.
You may assume substitution **f(a) = {a, a'},** and homomorphisms **g(a) = a'** and
h(a) = a, h(a') = λ. Here **a∈Σ** and **a'** is a new character associated with each such **a∈Σ**.
You only need give me the definition of **L||R** in an expression that obeys the above closure properties; you do not need to prove or even justify your expression.

L||R = ____

4 **4.** Specify True (**T**) or False (**F**) for each statement.

Statement	**T or F**
An algorithm exists to determine if a Phrase Structured Grammar generates λ	
If **P** is Unsolvable then Rice's Theorem can always show this	
The Context Sensitive Languages are closed under complement	
If **P** $\leq_m$ **Halt** then **P** must be RE	
The **RE** sets are closed under intersection	
The correct traces of a Turing Machine's Computations form a Context Free Language	
The Post Correspondence Problem is decidable if **|Σ| = 1**	
There is an algorithm to determine if **L** is finite, for **L** a Context Sensitive Language	

4 **5.** Let $\mathbf{P = \langle\langle x_1,x_2,\ldots,x_n\rangle, \langle y_1,y_2,\ldots,y_n\rangle\rangle}$, $\mathbf{x_i,y_1 \in \Sigma^+}$, $\mathbf{1 \le i \le n}$, be an arbitrary instance of **PCP**. We can use **PCP's undecidability** to show the undecidability of the problem to determine if a **Context Free Grammar** is ambiguous. Present grammars, **G1** and **G2**, associated with an arbitrary instance of PCP, **P**, such that $\mathcal{L}(\mathbf{G1}) \cap \mathcal{L}(\mathbf{G2})$ is non-empty if and only if there is a solution to **P**.
Define $\mathbf{G1 = (\{X\}, \Sigma \cup \{ [i] \mid 1 \le i \le n \}, R1, X)}$, $\mathbf{G2 = (\{Y\}, \Sigma \cup \{ [i] \mid 1 \le i \le n \}, R1, Y)}$,
where **R1 and R2** are the sets of rules (this is your job):

12 **6**. Choosing from among **(REC) recursive**, **(RE) re non-recursive, (coRE) co-re non-recursive**, **(NRNC) non-re/non-co-re**, categorize each of the sets in a) through d). Justify your answer by showing some minimal quantification of some known recursive predicate.

a.) A = { f | range(φ

a.) $\mathbf{A = \{ f \mid \text{range}(\varphi_f) \text{ has no values greater than } 10 \}}$

____ __________

b.) $\mathbf{B = \{ \langle f, x\rangle \mid \varphi_f \text{ converges on every input greater than or equal to } x \}}$

____ __________

c.) $\mathbf{C = \{ f \mid \varphi_f \text{ converges for at least one input of } x \text{ in at most } x \text{ steps} \}}$

____ __________

d.) $\mathbf{D = \{ f \mid \text{if } \varphi_f(f) \text{ converges it takes more than } f \text{ steps to do so} \}}$

____ __________

2 **7.** Looking back at Question 6, which of these are candidates for using Rice's Theorem to show their unsolvability? Check all for which Rice Theorem might apply.

a) ____ **b)** ____ **c)** ____ **d)** ____

8. Show that a set **S** is an infinite decidable (solvable/recursive) set if and only if it can be described as the range of a monotonically increasing algorithm. I will start the proof.

3 **a.)** Let **S** be an infinite recursive set. As **S** is decidable, it has a characteristic function χ_S where $\chi_S(x) = 1$, when $x \in S$, and $\chi_S(x) = 0$, otherwise. Using χ_S as a basis, we wish to define a monotonically increasing algorithm f_S whose range is **S**. Note that, since **S** is non-empty, it has a smallest element and, since it is infinite, it has no largest element. I have started the proof using primitive recursion (induction). You must complete it by writing in the formula to compute $f_S(y+1)$ given we know the value of $f_S(y)$.

Let $x \in S \Leftrightarrow \chi_S(x)$
// list the smallest element
Define $f_S(0) = \mu\, x\ \chi_S(x)$
// list the next item in monotonically increasing order. That's your job!!
$f_S(y+1) =$ ____

3 **b.)** Let **S** be the range of some monotonically increasing enumerating algorithm f_S. Show that **S** must be an infinite recursive set. First **S** is infinite since $\forall x\ f_S(x+1) > f_S(x)$. You must now present a characteristic function χ_S that takes advantage of the infinite nature of **S** and the fact that f_S is monotonically increasing and so enumerates any item **x** in some known bounded amount of time.

$\chi_S(x) =$ ____

6 **9**. Let sets **A** be a **non-empty** recursive (decidable) set and let **B** be re non-recursive (undecidable). Consider $C = \{ z \mid z = y^x$, where $x \in A$ and $y \in B \}$.
Note: Here, we define 0^0 to be **1** (yeah, I know that's a point of debate in Mathematics, but not in this question). For (a)-(c), either show sets **A** and **B** and the resulting set **C**, such that **C** has the specified property (argue convincingly that it has the correct property) or demonstrate (prove by construction) that this property cannot hold.

a. Can **C** be recursive? Circle **Y** or **N.**

b. Can **C** be re non-recursive? Circle **Y** or **N.**

c. Can **C** be non-re? Circle **Y** or **N.**

10. Define **CounterID (CI) = { f |** for all input **x**, **f(x) ≠ x }**.

2 **a.)** Show some minimal quantification of some known recursive predicate that provides an upper bound for the complexity of this set. (Hint: Look at **c.)** and **d.)** to get a clue as to what this must be.)

5 **b.)** Use Rice's Theorem to prove that **CI** is undecidable.

5 **c.)** Show that **TOTAL $\leq_m$ CI**, where **TOTAL = { f | ∀x f(x)↓ }**.

1 **d.)** From **a.)** through **c.)** what can you conclude about the complexity of **CI (Recursive, RE, RE-COMPLETE, CO-RE, CO-RE-COMPLETE, NON-RE/NON-CO-RE)**?