COT 5310 Fall 2006 More Final Exam Samples Name: *KEY*

12 **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 = Σ* ?	***D***	***U***	***U***
L = φ ?	***D***	***D***	***U***
L = L^2 ?	***D***	***U***	***U***
x ∈ L^2, for arbitrary x ?	***D***	***D***	***D***

8 **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?	***Y***	***N***
Closed under quotient?	***Y***	***N***
Closed under quotient with Regular languages?	***Y***	***Y***
Closed under complement?	***Y***	***N***

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

L_1 = { $a^i b^j c^k$ | $k \le i$ or $k \le j$ }

max(L_1) = *{ $a^i b^j c^k$ | k = max(i, j) }*

min(L_1) = *{ λ } (string of length 0)*

L_2 = { $a^i b^j c^k$ | $k > i$ or $k > j$ }

max(L_2) = *{ } (empty)*

min(L_2) = *{ $a^i b^j c^k$ | k = min(i, j)+1 }*

12 **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,
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.

Define the alphabet $\Sigma' = \{ a' \mid a \in \Sigma \}$***, where, of course,*** a' ***is a "new" symbol, i.e., one not in*** Σ.

Define homomorphisms g and h, and substitution f as follows:
$g(a) = a' \quad \forall a \in \Sigma \qquad h(a) = a \ ; \ \ h(a') = \lambda \ \ \forall a \in \Sigma \qquad f(a) = \{a, a'\} \quad \forall a \in \Sigma$

Consider $R = \Sigma^* \bullet g(\Sigma^*) \bullet \Sigma^* = \{ x\, y'\, z \mid x, y, z \in \Sigma^* \text{ and } y' = g(y) \in \Sigma'^* \}$
Σ^* ***is regular since it is the Kleene star closure of a finite set.***
$g(\Sigma^*)$ ***is regular since it is the homomorphic image of a regular language.***
R is regular as it is the concatenation of regular languages.

Now, $f(L) = \{ f(w) \mid w \in L \}$ ***is in C since C is closed under substitution. This language is the set of strings in L with randomly selected letters primed. Any string*** $w \in L$ ***gives rise to*** $2^{|w|}$ ***strings in f(L).***

$f(L) \cap R = \{ x\, y'\, z \mid x\, y\, z \in L \text{ and } y' = g(y) \}$ ***is in C since C is closed under intersection with regular languages.***

MissingMiddle(L) $= h(\ f(L) \cap R\ ) = \{ x\, z \mid \exists y \in \Sigma^* \text{ such that } xyz \in L \}$ ***which is in C, since C is closed under homomorphism. Q.E.D.***

10 **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 $\mathbf{G_A}$ and $\mathbf{G_B}$ based on some instance $\mathbf{P = \langle\langle x_1,x_2,\ldots,x_n\rangle, \langle y_1,y_2,\ldots,y_n\rangle\rangle}$ of **PCP**, such that $\mathbf{L(G_A) \cap L(G_B) \neq \phi}$ 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.

$G_A = (\ \{ A \}\ ,\ \Sigma \cup \{ [\ i\ ] \mid 1 \le i \le n \}\ ,\ A\ ,\ P_A\ ) \qquad G_B = (\ \{ B \}\ ,\ \Sigma \cup \{ [\ i\ ] \mid 1 \le i \le n \}\ ,\ B\ ,\ P_B\ )$

$P_A : A \to x_i\, A\, [\ i\ ] \mid x_i\, [\ i\ ] \qquad P_B : A \to y_i\, B\, [\ i\ ] \mid y_i\, [\ i\ ]$

$L(G_A) = \{ x_{i_1}\, x_{i_2} \ldots x_{i_p}\, [i_p] \ldots [i_2]\, [i_1] \mid p \ge 1,\ 1 \le i_t \le n,\ 1 \le t \le p \}$

$L(G_B) = \{ y_{j_1}\, y_{j_2} \ldots y_{j_q}\, [j_q] \ldots [j_2]\, [j_1] \mid q \ge 1,\ 1 \le j_u \le n,\ 1 \le u \le q \}$

$L(G_A) \cap L(G_B) = \{ w\, [k_r] \ldots [k_2]\, [k_1] \mid r \ge 1,\ 1 \le k_v \le n,\ 1 \le v \le r \}$***, where***

$$w = x_{k_1} x_{k_2} \ldots x_{k_r} = y_{k_1} y_{k_2} \ldots y_{k_r}$$

If $L(G_A) \cap L(G_B) \neq \phi$ ***then such a w exists and thus*** $k_1, k_2, \ldots, k_r$ ***is a solution to this instance of PCP. This shows that a decision procedure for the non-emptiness of the intersection of CFLs implies a decision procedure for PCP, which we have already shown is undecidable. Hence, the non-emptiness of the intersection of CFLs is undecidable. Q.E.D.***

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

First, show CONSTANT is non-trivial.
$C_0(x) = 0$, one of the base functions, is in CONSTANT
$S(x) = x+1$, one of the base functions, is not in CONSTANT
Thus, CONSTANT is non-trivial

Second, let f, g be two arbitrary partial recursive functions with the same I/O behavior.
That is, ∀x if f(x) is defined, then f(x) = g(x); otherwise both diverge, i.e., f(x)↑ and g(x)↑
Now, f ∈ CONSTANT
⇔ ∃K ∀x [f(x) = K] ***by definition of CONSTANT***
⇔ ∀x [g(x) = C] ***where C is the instance of K above, since ∀x [f(x) = g(x)]***
⇔ ∃K ∀x [g(x) = K] ***from above***
⇔ g ∈ CONSTANT ***by definition of CONSTANT***

Since CONSTANT meets both conditions of Rice's Theorem, it is undecidable. Q.E.D.

10 **7.** Show that **CONSTANT $\equiv_m$ TOT**, where **TOT = { f | ∀y φ_f(y)↓ }**.

CONSTANT $\le_m$ TOT
Let f be an arbitrary partial recursive function.
Define g_f using composition, primitive recursion and minimization by
$g_f(0) = f(0)$
$g_f(y+1) = f(y+1) + \mu z\ [f(y+1) = f(y)]$
Now, if f ∈ CONSTANT then ∀y [f(y)↓ and [f(y+1) = f(y)]].
Under this circumstance, $\mu z\ [f(y+1) = f(y)]$ is 0 for all y and $g_f(y) = f(y)$ for all y.
Clearly, then $g_f \in$ TOT
If, however, f ∉ CONSTANT then ∃y [f(y)↑ or [f(y+1) ≠ f(y)]].
Choose the least y meeting this condition.
If f(y)↑ then $g_f(y)$↑ since f(y) is in $g_f(y)$'s definition (the 1st term).
If f(y)↓ but [f(y+1) ≠ f(y)]] then $g_f(y)$↑ since $\mu z\ [f(y+1) = f(y)]$↑ (the 2nd term).
Clearly, then $g_f \notin$ TOT
Combining these, f ∈ CONSTANT ⇔ $g_f \in$ TOT and thus CONSTANT $\le_m$ TOT

TOT $\le_m$ CONSTANT
Let f be an arbitrary partial recursive function.
Define g_f using composition by
$g_f(y) = f(y) - f(y)$
Now, if f ∈ TOT then ∀y [f(y)↓] and thus ∀y $g_f(y) = 0$. Clearly , then $g_f \in$ CONSTANT
If, however, f ∉ TOT then ∃y [f(y)↑] and thus, ∃y [$g_f(y)$↑]. Clearly , then $g_f \notin$ CONSTANT
Combining these, f ∈ TOT ⇔ $g_f \in$ CONSTANT and thus TOT $\le_m$ CONSTANT

Hence, CONSTANT $\equiv_m$ TOT. Q.E.D.

8 **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 | $\forall$y φ_f(y) converges in no fewer than y steps }.**

We can deny the 2[nd] condition of Rice's Theorem since

S, where S(x) = x+1, converges in one step and hence is not in AT-LEAST-LINEAR

S', defined by primitive recursion, is in AT-LEAST-LINEAR, where

$S'(x) = C_1$

$S'(y+1) = S(S'(y))$

However, $\forall x$ [S(x) = S'(x)], so they have the same I/O behavior and yet one is in and the other is out of -LEAST-LINEAR, denying the 2[nd] condition of Rice's Theorem

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

We can deny the 1[st] condition of Rice's Theorem since all functions have an imposter. To see this, consider for any function f, the equivalent but distinct function $g(x) = f(x) + C_0(x)$. Thus, HAS-IMPOSTER is trivial since it is equal to $\aleph$, the set of all indices.

8 **9.** The trace language of a computational device like a Turing Machine is a language of the form **Trace = { $C_1\#C_2\# \ldots C_n\#$ | $C_i \Rightarrow C_{i+1}$, $1 \leq i < n$ }**

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**.

It is possible to create a Context Free Grammar for the language L = { C#C'# | ~ C $\Rightarrow$ C' }

Here, ~ C $\Rightarrow$ C' means C' is not derived directly from C.

The reason this is Context Free is that it checks just one pair, meaning we can push the first onto a stack and then compare the second to be sure it is not a consequence of the first.

Assume that the grammar for L is G_L = (V, N, T, P_L)

We can then extend this grammar to another CFG, G_{BAD} = (V $\cup$ {S, U, C}, N, S, P_{BAD}), where

P_{BAD} contains all of P_L plus

$$S \rightarrow U\,T\,U$$
$$U \rightarrow C\,\#\,U \mid \lambda$$

And C is a non-terminal that generates an arbitrary configuration in our machine notation.

The language BadTrace generated by G_{BAD} is then the strings that look like traces but have at least one error, i.e., one pair of configurations that does not reflect a correct step of computation. R $\cup$ BadTrace is the complement of Trace and is a CFL since it is the union of two CFLs (one is actually regular). Q.E.D.