Final Exam Topics 1

- Regular languages
  - Decision Problems
    - Membership
    - Emptiness
    - Finiteness
    - Σ^*
    - Equality
    - Containment
  - Closure
    - Union/Concatenation/Star
    - Complement
    - Substitution/Quotient, Prefix, Infix, Suffix
    - Max/Min

Final Exam Topics 2

- Context free languages
  - Writing a simple CFG
  - Decision Problems
    - Membership
    - Emptiness
    - Finiteness
    - Σ* (undecidable)
    - Equality (undecidable)
    - Containment (undecidable)
  - Closure
    - Union/Concatenation/Star
    - Intersection with Regular
    - Substitution/Quotient with Regular, Prefix, Infix, Suffix
  - Non-closure
    - intersection, complement, quotient, Max/Min
  - Pumping Lemma for CFLs

Final Exam Topics 3

- Chomsky Hierarchy
  (Red involve no constructive questions)
  - Regular, CFG, CSG, PSG (type 3 to type 0)
  - FSAs, PDAs, LBAs, Turing machines
  - Length preservation or increase makes membership in associated languages decidable for all but PSGs
  - CFLs can be inherently ambiguous but that does not mean a language that has an ambiguous grammar is automatically inherently ambiguous

Final Exam Topics 4

- Computability Theory
  - Decision problems: solvable (decidable, recursive), semi-decidable (recognizable, recursively enumerable/re, generable), non-re
  - A set is re iff it is semi-decidable
  - If set is re and complement is also re, set is recursive (decidable)
  - Halting problem (K_0): diagonalization proof of undecidability
    - Set K_0 is re but complement is not
  - Set K = { f | f(f) converges }
  - Algorithms (Total): diagonalization proof of non-re
  - Reducibility to show certain problems are not decidable or even non-re
  - K and K_0 are re-complete – reducibility to show these results
  - Rice's Theorem: All non-trivial I/O properties of functions are undecidable (weak and strong versions)
  - Use of quantification to discover upper bound on complexity

Final Exam Topics 5

- Computability Applied to Formal Grammars
  (Red only results not constructions that lead to these)
  - Post Correspondence problem (PCP)
    - Definition
    - Undecidability (proof was only sketched and is not part of this course)
    - Application to ambiguity and non-emptiness of intersections of CFLs and to non-emptiness of CSLs
  - Traces of Turing computations
    - Not CFLs
    - Single steps are CFLs (use reversal of second configuration)
    - Intersections of pairwise correct traces are traces
    - Complement of traces (including terminating traces) are CFL
    - Use to show cannot decide if CFL, L, is Σ^*
    - L= Σ^* and L = L^2 are undecidable for CFLs
  - PSG can mimic TM, so generate any re language; thus, membership in PSL is undecidable, as is emptiness of PSL.
  - All re sets are homomorphic images of CSLs (erase fill character)

Final Exam Topics 6

- Complexity Theory
  - Verifiers versus solvers: P versus NP
  - Definitions of NP: verify in deterministic poly time vs solve in non-deterministic polynomial time
  - Co-P and co-NP; NP-Hard versus NP-Complete
  - Basic idea behind SAT as NP-Complete
  - Reduction of SAT to 3-SAT to Subset-Sum
  - Equivalence of Subset-Sum to Partition
  - Relation of Subset-Sum and Partition to multiprocessor scheduling
  - Vertex cover, 3-coloring, register allocation, Independent set
  - Gadgets for above

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.) $D = \sim C$ **RE, NR**

b.) $D \subseteq (A \cup C)$ **REC, RE, NR**

c.) $D = \sim B$ **NR**

d.) $D = B - A$ **REC, RE**

2. Prove that the **Halting Problem** (the set $\mathbf{K_0}$ **)** is not recursive (decidable) within any formal model of computation. (Hint: A diagonalization proof is required here.)

Assume we can decide the halting problem. Then there exists some total function Halt such that

$$\text{Halt}(x,y) = \begin{cases} 1 & \text{if } [x]\,(y) \text{ is defined} \\ 0 & \text{if } [x]\,(y) \text{ is not defined} \end{cases}$$

Here, we have numbered all programs and [x] refers to the x-th program in this ordering. We can view Halt as a mapping from $\aleph$ into $\aleph$ by treating its input as a single number representing the pairing of two numbers via the one-one onto function

$$\text{pair}(x,y) = \langle x,y \rangle = 2^x\,(2y+1) - 1$$

with inverses $\langle z \rangle_1 = \exp(z+1,1)$ **and** $\langle z \rangle_2 = (((z+1) \,//\, 2^{\langle z \rangle_1}) - 1) \,//\, 2$

Now if Halt exist, then so does Disagree, where

$$\text{Disagree}(x) = \begin{cases} 0 & \text{if Halt}(x,x) = 0, \text{ i.e, if } \Phi_x(x) \text{ is not defined} \\ \mu y\,(y == y+1) & \text{if Halt}(x,x) = 1, \text{ i.e, if } \Phi_x(x) \text{ is defined} \end{cases}$$

Since Disagree is a program from $\aleph$ into $\aleph$, Disagree can be reasoned about by Halt. Let d be such that Disagree = Φ_d, then

Disagree(d) is defined $\Leftrightarrow$ Halt(d,d) = 0 $\Leftrightarrow$ Φ_d (d) is undefined $\Leftrightarrow$ Disagree(d) is undefined

But this means that Disagree contradicts its own existence. Since every step we took was constructive, except for the original assumption, we must presume that the original assumption was in error. Thus, the Halting Problem is not solvable.

3.Using reduction from the known undecidable **HasZero,**
HZ = { f | ∃x f(x) = 0 }, show the non-recursiveness (undecidability) of the problem to decide if an arbitrary recursive function **g** has the property **IsZero, Z = { f | ∀x f(x) = 0 }**.

HZ = { f | ∃x ∃t [STP(f, x, t) & VALUE(f, x, t) == 0] }
Let f be the index of an arbitrary effective procedure.
Define $g_f(y)$ = 1 - ∃x ∃t [STP(f, x, t) & VALUE(f, x, t) == 0]
If ∃x f(x) = 0, we will find the x and the run-time t, and so we will return 0 (1 – 1)
If ∀x f(x) ≠ 0, then we will diverge in the search process and never return a value.
Thus, f ∈ HZ iff g_f ∈ Z = { f | ∀x f(x) = 0 }.

4. 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**
$L = \Sigma^*$?	***D***	***U***
$L = \phi$?	***D***	***D***
$x \in L^2$, for arbitrary x ?	***D***	***D***

5. 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***

6. 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 | ∃y ∈ Σ* such that xyz ∈ L }
You must be very explicit, describing what is produced by each transformation you apply.

Define the alphabet Σ' = { a' | a∈Σ }, 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' ∀a∈Σ h(a) = a ; h(a') = λ ∀a∈Σ f(a) = {a, a' }
∀a∈Σ

Consider R = Σ* • g(Σ*) • Σ* = { x y' z | x, y, z ∈Σ* and y'=g(y) ∈Σ'* }
Σ* is regular since it is the Kleene star closure of a finite set.
g(Σ*) 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) | w ∈ 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∈L gives rise to $2^{|w|}$ strings in f(L).

f(L) ∩ R = { x y' z | x y z ∈ L and y'=g(y) } is in C since C is closed under intersection with regular languages.

MissingMiddle(L) = h(f(L) ∩ R) = { x z | ∃y ∈ Σ* such that xyz ∈ L } which is in C, since C is closed under homomorphism. Q.E.D.

7. 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 $P = \langle\langle x_1,x_2,\ldots,x_n\rangle, \langle y_1,y_2,\ldots,y_n\rangle\rangle$ of **PCP**, such that $L(G_A) \cap L(G_B) \neq \phi$ 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.

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

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

$L(G_A) = \{ x_{i1} \, x_{i2} \ldots x_{ip} \, [i_p] \ldots [i_2] \, [i_1] \mid p \ge 1, 1 \le i_t \le n, 1 \le t \le p \}$

$L(G_B) = \{ y_{j1} \, y_{j2} \ldots y_{jq} \, [j_q] \ldots [j_2] \, [j_1] \mid q \ge 1, 1 \le j_u \le n, 1 \le u \le q \}$

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

$$w = x_{k1} \, x_{k2} \ldots x_{kr} = y_{k1} \, y_{k2} \ldots y_{kr}$$

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.

8. Consider the set of indices **CONSTANT = { f | ∃K ∀y [$\varphi_f(y)$ = 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.

Z(x) = 0 is in CONSTANT

S(x) = x+1 is not in CONSTANT

Thus, CONSTANT is non-trivial

Second, let f, g be two arbitrary computable 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 the 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 the definition of CONSTANT**

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

9. Show that **CONSTANT** $\equiv_m$ **TOT**, where **TOT** = { f | $\forall y\ \varphi_f(y)\downarrow$ }.

CONSTANT $\leq_m$ **TOT**

Let f be an arbitrary effective procedure.

Define g_f 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 $\forall y$ [$f(y)\downarrow$ 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 $\exists y\ [f(y+1) \neq f(y)]$ or $\exists y\ f(y)\uparrow$.

Choose the least y meeting this condition.

If $f(y)\uparrow$ then $g_f(y)\uparrow$ since f(y) is in $g_f(y)$'s definition (the 1st term).

If $f(y)\downarrow$ but $[f(y+1) \neq f(y)]$ then $g_f(y)\uparrow$ since $\mu z\ [f(y+1) = f(y)]\uparrow$ (the 2nd term).

Clearly, then $g_f \notin$ TOT

Combining these, f ∈ CONSTANT ⇔ $g_f \in$ TOT and thus CONSTANT $\leq_m$ **TOT**

TOT $\leq_m$ CONSTANT

Let f be an arbitrary effective procedure.

Define g_f by $g_f(y) = f(y) - f(y)$

Now, if $f \in$ TOT then $\forall y$ [$f(y)\downarrow$] and thus $\forall y\ g_f(y) = 0$.

Clearly, then $g_f \in$ CONSTANT

If, however, $f \notin$ TOT then $\exists y$ [$f(y)\uparrow$] and thus, $\exists y$ [$g_f(y)\uparrow$]. Clearly, then $g_f \notin$ CONSTANT

Combining these, $f \in$ TOT $\Leftrightarrow$ $g_f \in$ CONSTANT and thus

TOT $\leq_m$ CONSTANT

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

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

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

Z, where Z(x) = 0, implemented by the TM R converges in one step no matter what x is and hence is not in AT-LEAST-LINEAR

Z', defined by TM $\mathcal{L}$ $\mathcal{R}$ R, is in AT-LEAST-LINEAR since it takes over 2*|input| steps.

However, $\forall x$ [Z(x) = Z'(x)], so they have the same I/O behavior and yet one is in and the other is out of AT-LEAST-LINEAR, denying the 2nd 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 1st 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) + 0. Thus, HAS-IMPOSTER is trivial since it is equal to $\aleph$, the set of all indices.

11. We described the proof that **3SAT** is polynomial reducible to Subset-Sum.

a.) Describe **Subset-Sum**

Given a sequence of n positive integers, <i1,…in> and a goal, G, which is also a positive integer, is there a subset of the integers from the sequence that sums to the goal value?

b.) Show that **Subset-Sum** is in **NP**

Give a proposed solution we can check if its sum equals G in linear time. Any decision problem where a solution can be verified in linear time is in NP, so we are done.

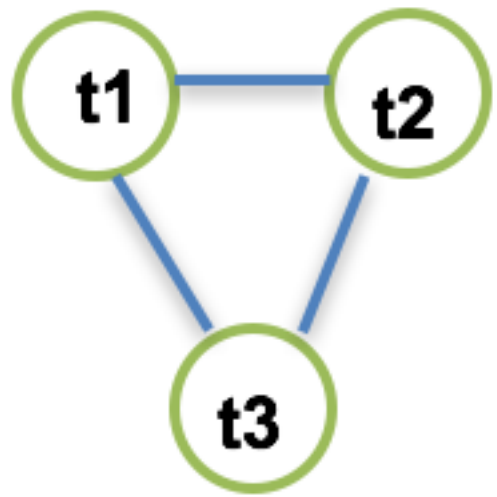
c.) Assuming a **3SAT** expression **(a + ~b + c) (b + b + ~c)**, fill in the upper right part of the reduction from **3SAT** to **Subset-Sum**.

	a	b	c	a + ~b + c	b + b + ~c
a	**1**	**0**	**0**	**1**	**0**
~a	**1**	**0**	**0**	**0**	**0**
b	**0**	**1**	**0**	**0**	**1 or 2**
~b	**0**	**1**	**0**	**1**	**0**
c	**0**	**0**	**1**	**1**	**0**
~c	**0**	**0**	**1**	**0**	**1**
C1	**0**	**0**	**0**	**1**	**0**
C1'	**0**	**0**	**0**	**1**	**0**
C2	**0**	**0**	**0**	**0**	**1**
C2'	**0**	**0**	**0**	**0**	**1**
	1	**1**	**1**	**3**	**3**

12. Describe the gadgets used to reduce 3SAT to the Vertex Covering Problem

Variable Gadgets

Clause Gadgets



13. Show a first-fit schedule for the following task times on two processors

{T1/1, T2/7, T3/2, T4/4, T5/4, T6/2, T7/5, T8/2, T9/3, T10/4}

T1	**T3**	**T3**	**T4**	**T4**	**T4**	**T4**	**T5**	**T5**	**T5**	**T5**	**T8**	**T8**	**T9**	**T9**	**T9**		
T2	**T2**	**T2**	**T2**	**T2**	**T2**	**T2**	**T6**	**T6**	**T7**	**T7**	**T7**	**T7**	**T7**	**T10**	**T10**	**T10**	**T10**

14. Use the Pumping Lemma for CFLs to show:
{ ww | w is over {a,b} } is not Context Free

Assume the language L = { ww | w is over {a,b} } is Context Free. Let N>0 be the value associated with L by the Pumping Lemma for Context Free languages.
$a^N b^N a^N b^N \in L$.

By Pumping Lemma, $a^N b^N a^N b^N = uvwxy$, for some strings u,v,w,x,y over {a,b}, where $|vx| > 0$, $|vwx| \leq N$ and $\forall i \geq 0$ $uv^i wx^i y \in L$.

All cases collapse into the following analysis. vwx must include at most one of the 'a' sequences and at most one of the 'b' sequences; moreover it must have at least one of these cases (first 'a' sequence but not second; first 'b' sequence but not second; second 'a' sequence but not first; or second 'b' sequence but not first). Set i=0 and we have removed letters from one of the 'a' sequences and/or one of the 'b' sequences, but not the other. This denies that uwy is in L, thereby contradicting the Pumping Lemma.

15. Write a context-free grammar for the complement of

{ ww | w is over {a,b} }

S → L<Odd> | AB | BA

<Odd> → L<Even> | λ

<Even> → L<Odd>

A → L A L | a

B → L B L | b

L → a | b

Sample Question#5

5. Let **S** be an re (recursively enumerable), non-recursive set, and **T** be an re, possibly recursive set. Let

   E = { z | z = x + y, where x ∈ S and y ∈ T }.

   Answer with proofs, algorithms or counterexamples, as appropriate, each of the following questions:

   (a) Can **E** be non re? No. If T = ∅ then E is recursive.
   Assume S is non-empty and S and T are enumerated by f_S, f_T, resp.
   Then $f_E(<x,y>) = f_S(x) + f_S(y)$ enumerates E.

   (b) Can **E** be re non-recursive? Yes. T = {0}, E = S

   (c) Can **E** be recursive? Yes, T=ℵ, E = { x | x ≥ min value in S }

Assignment # 8.1 Key

1. Use reduction from Halt to show that one cannot decide REPEATS, where
   REPEATS = { f | for some x and y, $x \neq y$, $\varphi_f(x)\downarrow$, $\varphi_f(y)\downarrow$ and $\varphi_f(x) == \varphi_f(y)$ }

Let f,x be an arbitrary pair of natural numbers. <f,x> is in Halt iff $\varphi_f(x)\downarrow$

Define g by $\varphi_g(y) = \varphi_f(x) - \varphi_f(x)$, for all y.

Clearly, $\varphi_g(y) = 0$, for all y, iff $\varphi_f(x)\downarrow$, and $\varphi_g(y)\uparrow$, for all y, otherwise.

Summarizing, <f,x> is in Halt implies g is in REPEATS and <f,x> is not in Halt implies g is not in REPEATS

Halt $\leq_m$ REPEATS as we were to show.

Note: I have not overloaded the index of a function with the function in my proof, but I do not mind if you do such overloading.

Assignment # 8.2 Key

2. Show that REPEATS reduces to Halt. (1 plus 2 show they are equally hard)

Let f be an arbitrary natural number. f is in REPEATS iff for some x and y, $x \neq y$, $\varphi_f(x)\downarrow$, $\varphi_f(y)\downarrow$ and $\varphi_f(x) == \varphi_f(y)$

Define g by $\varphi_g(z) = \exists \langle x,y,t \rangle$ [STP(f,x,t) & STP(f,y,t) & (x≠y) & (VALUE(f,x,t) = (VALUE(f,y,t))], for all z.

Clearly, $\varphi_g(z) = 1$, for all z, iff there is some pair, x,y, such that $\varphi_f(x)\downarrow$ and $\varphi_f(y)\downarrow$ and $\varphi_f(x) = \varphi_f(y)$, and $\varphi_g(z)\uparrow$, for all z, otherwise.

Summarizing, f is in REPEATS iff g is in Halt and so

REPEATS $\leq_m$ Halt as we were to show.

Assignment # 8.3 Key

3. Use Reduction from Total to show that DOUBLES is not even re, where
   DOUBLES = { f | for all x, $\varphi_f(x)\downarrow$, $\varphi_f(x+1)\downarrow$ and $\varphi_f(x+1)=2*\varphi_f(x)$ }

Let f be an arbitrary natural number. f is in Total iff $\forall x\ \varphi_f(x)\downarrow$

Define g by $\varphi_g(x) = \varphi_f(x) - \varphi_f(x)$, for all x.

Clearly, $\varphi_g(x) = 0$, and so $\varphi_g(x+1) = 2*\varphi_g(x) = 0$ for all x, iff $\forall x\ \varphi_f(x)\downarrow$; otherwise $\varphi_g(x)\uparrow$ for some x.

Summarizing, f is in Total iff g is in DOUBLES and so

TOTAL $\leq_m$ DOUBLES as we were to show.

Assignment # 8.3 Alternate Key

3. Use Reduction from Total to show that DOUBLES is not even re, where
   DOUBLES = { f | for all x, $\varphi_f(x)\downarrow$, $\varphi_f(x+1)\downarrow$ and $\varphi_f(x+1)=2*\varphi_f(x)$ }

Let f be an arbitrary natural number. f is in Total iff $\forall x\ \varphi_f(x)\downarrow$

Define g by $\varphi_g(x) = \varphi_f(x) - \varphi_f(x)+2\text{^}x$ for all x.

Clearly, $\varphi_g(x) = 2\text{^}x$, and so $\varphi_g(x+1) = 2*\varphi_g(x) = 2\text{^}(x+1)$ for all x, iff $\forall x$ $\varphi_f(x)\downarrow$; otherwise $\varphi_g(x)\uparrow$ for some x.

Summarizing, f is in Total iff g is in DOUBLES and so

TOTAL $\leq_m$ DOUBLES as we were to show.

Assignment # 8.4 Key

4. Show **DOUBLES** reduces to **Total**. (4 plus 5 show they are equally hard)

Let f be an arbitrary natural number. f is in DOUBLES iff $\forall x\ \varphi_f(x)\downarrow$, $\varphi_f(x+1)\downarrow$ and $\varphi_f(x+1)=2*\varphi_f(x)$.

Define g by $\varphi_g(x) = \mu y[\varphi_f(x+1) = 2*\varphi_f(x)]$, for all x.

Clearly, $\varphi_g(x)\downarrow$, for all x, iff $\forall x\ \varphi_f(x)\downarrow$, $\varphi_f(x+1)\downarrow$ and $\varphi_f(x+1)=2*\varphi_f(x)$; otherwise $\varphi_g(x)\uparrow$ for some x.

Summarizing, f is in DOUBLES iff g is in Total and so

DOUBLES $\leq_m$ **TOTAL** as we were to show.

Assignment # 8.5 Key

5. Use Rice's Theorem to show that REPEATS is undecidable

First, REPEATS is non-trivial as C0(x) = 0 is in REPEATS and S(x) = x+1 is not.

Second, REPEATS is an I/O property.

To see this, let f and g are two arbitrary indices such that $\forall x\,[\varphi_f(x) = \varphi_g(x)]$

f $\in$ REPEATS iff $\exists$ y,z, y ≠ z, such that $\varphi_f(y)\downarrow$, $\varphi_f(z)\downarrow$ and $\varphi_f(y) = \varphi_f(z)$ iff, since $\forall x\,[\varphi_f(x) = \forall x\,\varphi_g(x)]$, $\exists$ y,z, y ≠ z, y ≠ z, (same y,z as above) such that $\varphi_g(y)\downarrow$, $\varphi_g(z)\downarrow$ and $\varphi_g(y) = \varphi_g(z)$ iff g $\in$ REPEATS.

Thus, f $\in$ REPEATS iff g $\in$ REPEATS.

Assignment # 8.6 Key

6. Use Rice's Theorem to show that DOUBLES is undecidable

First, DOUBLES is non-trivial as C0(x) = 0 (2*0 = 0) is in DOUBLES and S(x) = x+1 is not.

Second, DOUBLES is an I/O property.

To see this, let f and g are two arbitrary indices such that $\forall x\,[\varphi_f(x) = \varphi_g(x)]$.

$f \in$ DOUBLES iff for all x, $\varphi_f(x)\downarrow$, $\varphi_f(x+1)\downarrow$ and $\varphi_f(x+1)=2*\varphi_f(x)$ iff, since $\forall x\,[\varphi_f(x) = \varphi_g(x)]$, for all x, $\varphi_g(x)\downarrow$, $\varphi_g(x+1)\downarrow$ and $\varphi_g(x+1)=2*\varphi_g(x)$ iff $g \in$ DOUBLES.

Thus, $f \in$ DOUBLES iff $g \in$ DOUBLES.

Assignment # 9.1a Key

1. Use quantification of an algorithmic predicate to estimate the complexity (decidable, re, co-re, non-re) of each of the following, (a)-(d):

a)REPEATS = { f | for some x and y, x ≠ y, f(x)↓, f(y)↓ and f(x) == f(y) }

∃<x,y,t>[STP(f,x,t) & STP(f,y,t) & (x≠y) & (VALUE(f,x,t) = (VALUE(f,y,t))]

RE

Assignment # 9.1b Key

b) DOUBLES = { f | for all x, f(x)↓, f(x+1)↓ and f(x+1)=2*f(x) }

$\forall x \exists t$ [STP(f,x,t) & STP(f,x+1,t) & (2*VALUE(f,x,t) = (VALUE(f,x+1,t))]

Non-RE, Non-Co-RE

Assignment # 9.1c Key

c) DIVEVEN = { f | for all x, f(2*x)↑ }

∀<x,t> [~STP(f,2*x,t)]

Co-RE

Assignment # 9.1d Key

d) QUICK10={ f | f(x), for all 0≤x≤9, converges in at most x+10 steps }

STP(f,0,10) & STP(f,1,11) & … & STP(f,9,19)

or

$\forall x_{0\le x\le 9}$ [STP(f,x,x+10)]

REC

Assignment # 9.21 Key

1. Let sets A be recursive (decidable) and B be re non-recursive (undecidable).
   Consider C = { z | min(x,y), where x ∈ A and y ∈ B }. For (a)-(c), either show sets A and B with the specified property or demonstrate that this property cannot hold.

a) Can C be recursive?

YES. Consider A = {0}. B = Halt. C = {0}

Assignment # 9.2b Key

b) Can C be non-recursive?

YES. Consider A = { 2x | x ∈ *N* }. B = { 2x+1 | x ∈ Halt}. C = A ∪ B. This is semi-decidable but non re as Halt is reducible to C.

Assignment # 9.2c Key

c) Can C be non-re?

No. Can enumerate C as follows.

First if A is empty then C is empty and so RE by definition.

If A is non-empty then A is enumerated by some algorithm f_A as recursive sets are RE.

As B is non-recursive RE, then it is non-empty and enumerated by some algorithm f_B.

Define f_C by $f_C(<x,y>) = \min(f_A(x),f_B(y))$. f_C is clearly an algorithm as it is the composition of algorithms. The range of f_C is then $\{ z \mid \min(x,y)$, where $x \in A$ and $y \in B \} = C$ and so C must be RE.