Computer Science Qualifying Exam
Foundations January 1995

1. **Question.** Prove or disprove that the following language is regular, where the alphabet is $\sum = \{a, b, \#\}$.

$$L = \{x\#y | x, y \in (a+b)^* \text{ and } y \text{ is a prefix of } x\}$$

Solution. L is *not* regular by the Pumping Lemma.

Proof: Assume L is regular. Let n be the P.L. constant. Pick $z = z^n b\#a^n b$.

Adversary divides $z = uvw$ such that $|uv| \leq n$ and $k = |v| \geq 1$. Then $v \in a^+$ because $|uv| \leq n$.

By the P.L., $uv^2w = a^{n+k}b\#a^n b$ is not a prefix of $a^{n+k}b$, so $uv^2w \notin L$. This contradicts the Pumping Lemma, so L is not regular.

2. **Question.** Give a context-*sensitive* grammar for the following language, where the alphabet is $\sum = \{a, b, \#\}$.

$$L = \{x\#x | x \in (a+b)^*\}$$

Solution.

$S \rightarrow A\#a$	$\mid$ $B\#b$	$\mid$ $\#$		$Xa \rightarrow aX$
$A \rightarrow AaX$	$\mid$ BaY	$\mid$ a		$Xb \rightarrow bX$
$B \rightarrow AbX$	$\mid$ BbY	$\mid$ b		$Ya \rightarrow aY$
$X\# \rightarrow \#a$				$Yb \rightarrow bY$
$Y\# \rightarrow \#b$				

Idea: The intermediate results look like $x\#x$, *except* that the leftmost character in x is a variable A or B for a or b. For example: $Aabab\#aabab$. This leftmost variable produces new characters in pairs: a with X together, and b and Y. Then, X or Y migrates to the right until it hits the $\#$ and produces a terminal.

Sample derivation:

$$S \rightarrow B\#b \rightarrow BbY\#b \rightarrow AbXbY\#b \rightarrow abXbY\#b \rightarrow abbXY\#b \rightarrow abbX\#bb \rightarrow abb\#abb$$

Alternate Solution.

$S \rightarrow SAa$	$\mid$ SBb	$\mid$ $\#$
$aA \rightarrow Aa$	$bA \rightarrow Ab$	
$aB \rightarrow Ba$	$bB \rightarrow Bb$	
$\#A \rightarrow a\#$	$\#B \rightarrow b\#$	

Note. The rules of a context-sensitive grammar require that $|\alpha| \leq |\beta|$ in every production $\alpha \rightarrow \beta$. In particular, the production $A \rightarrow \epsilon$ is *not* allowed.

3. **Question.** Construct a partial recursive function $f : N \to N$ such that (i) f is defined infinitely often and (ii) f grows faster than any total recursive function (meaning that there is not total recursive g such that $f(n) \leq g(n)$ for all n such that $f(n)$ is defined), and prove your answer.

   Solution. Let $\phi_1, \phi_2, \ldots$ be an effective enumeration of all partial recursive functions from N to N. Define

$$f(k) = \phi_k(k) + 1$$

   Claim 1: f is partial recursive because it is possible to effectively enumerate all partial recursive functions. Of course, if $\phi_k(k)$ is undefined, then $f(k)$ is also undefined.

   Claim 2: f is defined infinitely often. We know that $K = \{k | \phi_k(k) \text{ halts } \}$ is undecideable. If f is defined only finitely often, then K is finite and hence decideable.

   Claim 3: There is no total recurseive g that bounds f. If g is total recursive, then $g = \phi_k$ for some k, and further $\phi_k(k)$ is defined. But then $f(k) > g(k)$.

   Alternate Solution. Let $M_1, M_2, \ldots$ be an enumeration of all Turing machines. Define $f'(k)$ to be the number of moves M_k makes on input k, or else undefined if $M_k(k)$ does not halt. Claims 1–3 can also be proved for f'.

   Note. (1) It is *not* possible to choose $\phi_1, \phi_2, \ldots$ to be an enumeration of all *total* recursive functions. (2) It is *not* possible to make f be total recursive.

4. **Question.** In the propositional calculus, let $B = \{0, 1\}$ be the set of boolean values where $0 = false, 1 = true$ and $0 < 1$. A boolean function $f : B^n \to B$ is called *monotonic* if

$$f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \leq f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

   for all $x_1, \ldots, x_n \in B$ and for all $1 \leq i \leq n$. That is, flipping an input to f from 0 to 1 can never flip the output of f from 1 to 0.

   Prove that any monotonic boolean function (with any number of inputs) can be expressed using only the operators $\wedge$, $\vee$, and the constants 0, 1 (and of course the inputs $x_1, \ldots, x_n$).

   Hint: Consider $f(0, x_2, \ldots, x_n)$ and $f(1, x_2, \ldots, x_n)$.

   Solution. Proof by induction on n. Define the induction hypothesis as

$$P(n) \equiv \text{ any monotonic } f : B^n \to B \text{ can be expressed with } \wedge, \vee, 1, 0$$

   Base ($n = 0$). The only boolean functions with 0 inputs are $f = 0$ and $f = 1$, and both of these are in the proper form.

Base ($n = 1$). The only boolean functions with 1 input are $f(x) = 0$, $f(x) = 1$, $f(x) = x$, and $f(x) = \neg x$. The first three are in the proper form and the last one is not monotonic.

Ind ($n \geq 2$). Let $f : B^n \to B$ be monotonic. Then

$$f(x_1, x_2, \ldots, x_n) \equiv f(0, x_2, \ldots, x_n) \vee [x_1 \wedge f(1, x_2, \ldots, x_n)]$$

This can be verified by plugging in $x_1 = 0$ and $x_1 = 1$ into both sides of the above equation and using the property that $f(x_1, \ldots, x_n)$ is monotonic.

Further, both $f(0, x_2, \ldots, x_n)$ and $f(1, x_2, \ldots, x_n)$ are monotonic, which follows immediately from the fact that $f(x_1, \ldots, x_n)$ is monotonic.

By the induction hypothesis, $f(0, x_2, \ldots, x_n)$ and $f(1, x_2, \ldots, x_n)$ can be expressed with $\wedge, \vee, 1, 0$, and the above formula only uses $\wedge$ and $\vee$, so $f(x_1, \ldots, x_n)$ can be written with $\wedge, \vee, 1, 0$.

5. **Question**. Let D be a cpo (complete partial order) and let $D \to D$ be the cpo of continuous functions mapping D into D. Let $F, G \in D \to D$ such that F approximates G. Prove that the least fixpoint of F approximates the least fixpoint of G.

   Solution. Since F, G are monotonic, it is easy to prove by induction that for all $i \in N$, $F^i(\perp) \sqsubseteq F^{i+1}(\perp)$ and $G^i(\perp) \sqsubseteq G^{i+1}(\perp)$. Moreover, since $F \subseteq G$, we know that $F(x) \sqsubseteq G(x)$ for all $x \in D$. Hence, it is easy to prove by induction that for all $i \in N$ that

$$F^{i+1}(\perp) \sqsubseteq F(G^i(\perp)) \sqsubseteq G^{i+1}(\perp)$$

   .

   Since F and G are continuous, their least fixpoints are the least upper bounds of the chains $\{F^i(\perp) | i \in N\}$ and $\{G^i(\perp) | i \in N\}$, respectively. But the least upper bound of $\{G^i(\perp) | i \in N\}$ must be an upper bound for $\{F^i(\perp) | i \in N\}$ because every element of $\{F^i(\perp) | i \in N\}$ is bounded by a corresponding element of $\{G^i(\perp) | i \in N\}$ and the $\sqsubseteq$ relation is transitive. hence, the least upper bound of $\{F^i(\perp) | i \in N\}$ must approximate the least upper bound of $\{G^i(\perp) | i \in N\}$, which is what we were asked to prove.

6. **Question**. A term M in the λ-*calculus* has the form defined by the grammar

$$M \quad ::= \quad x | (\lambda x.M) | (MM)$$

   where x ranges over a set of variables. The β reduction relation $\to_\beta$ on λ-terms is the compatible closure of the relation determined by the rule

$$((\lambda x.M)N) \to_\beta M[N/x]$$

where $M[N/x]$ is the capture-avoiding substitution of N for the free occurrences of x in M.

A term M is in *normal form* if there is no term N such that $M \rightarrow_\beta N$.

(a) Find a λ-term M that has both an infinite reduction sequence

$$M \rightarrow_\beta M_1 \rightarrow_\beta M_2 \rightarrow_\beta \cdots \rightarrow_\beta M_k \rightarrow_\beta \cdots$$

and a normal form

$$M \rightarrow_\beta N_1 \rightarrow_\beta \cdots \rightarrow_\beta N_n$$

where N_n is in normal form. Show the infinite reduction sequence (at least the first few terms) and the reduction to normal form.

(b) Extend your solution in (a) to a λ-term M that has both an infinite *non-repeating* reduction sequence

$$M \rightarrow_\beta M_1 \rightarrow_\beta M_2 \rightarrow_\beta \cdots \rightarrow_\beta M_k \rightarrow_\beta \cdots$$

and a normal form

$$M \rightarrow_\beta N_1 \rightarrow_\beta \cdots \rightarrow_\beta N_n$$

An infinite reduction sequence

$$M_1 \rightarrow_\beta M_2 \rightarrow_\beta \cdots \rightarrow_\beta M_k \rightarrow_\beta \cdots$$

is non-repeating iff $M_i = M_j$ (syntactic identity) implies that $i = j$. Show the infinite reduction sequence and the reduction to normal form.

Solution.

(a) Let $M = (\lambda x.y)\Omega$ where $\Omega = ((\lambda x.xx)(\lambda x.xx))$. Clearly, $M \rightarrow_\beta y$ which is in normal form, yet

$$M \rightarrow_\beta M \rightarrow_\beta M \rightarrow_\beta \cdots$$

because $\Omega \rightarrow_\beta \Omega$.

(b) Let $M = (\lambda x.y)\Omega'$ where $\Omega = (\Theta\Theta)$ and $\Theta = (\lambda x.xxx)$. As above, $M \rightarrow_\beta y$ which is in normal form, yet

$$M \rightarrow_\beta M\Theta \rightarrow_\beta M\Theta\Theta \rightarrow_\beta \cdots$$

because $\Omega' \rightarrow_\beta \Omega'\Theta$.