CS 380C: Assignment 3
Problem set on Fixpoint Equations
Spring 2023

Due: 11:59 PM, February 26th, 2023

In this assignment, the word $domain$ refers to a finite set S with a partial order $\leq (\subseteq S \times S)$ under which there is a least element. We will write $D = (S, \leq)$ to represent the domain.

1. (Iterative solution of linear systems, 10 points) Consider the linear system

   $3x + y = 4$
   $x + 2y = 3$

   (a) (5 points) Write down the recurrence relation that corresponds to solving this system using the Jacobi method, starting with the initial approximation $(x_1 = 0, y_1 = 0)$. Use the first equation to refine the approximation for x and the second equation to refine the approximation for y. Express this recurrence as a computation involving matrices and vectors.

   (b) (2 points) Compute the first 10 approximations (x_i, y_i) and plot a 3D plot (x, y, i) in which the z-axis is the iteration number i. Give an intuitive explanation of this 3D plot. You do not need to turn in any code but turn in your plot and explanation. You can use Matlab, Octave, or any other program for the 3D plot.

   (c) (3 points) Repeat these two parts for the Gauss-Seidel method. You can find a description of the Gauss-Seidel method online.

2. (10 points) Consider the domain D that is the powerset of the set {a,b,c}, in which elements are ordered by subset ordering, and $\{\}$ is the least element.

   (a) (1 point) Draw a Hasse diagram of this partially ordered set. You do not have to show transitive edges.

   (b) (2 points) Write down a function D$\rightarrow$ D that is monotonic but not extensive.

   (c) (2 points) Write down a function D$\rightarrow$ D that is extensive but not monotonic.

(d) (2 points) Write down a function D→ D that is both extensive and monotonic.

(e) (2 points) Write down a function $f : D \to D$ for which the equation $x = f(x)$ has no solutions. Explain why your function is not monotonic.

(f) (1 point) Write down a function $f : D \to D$ for which the equation $x = f(x)$ has multiple solutions.

3. (10 points) Let $D = (S, \leq)$ be a domain, and let $f : D \to D$ be monotonic. Let set $L \subseteq S$ be the set of solutions to the equation $x = f(x)$. Show that $(L, \leq)$ is itself a domain.

4. (10 points) Let $D = (\mathscr{S}, \subseteq)$ be the domain consisting of the powerset of a set S ordered by subset ordering, and let $a, b \in S$. Consider the function $f : \mathscr{S} \to \mathscr{S}$ defined as follows: $f(T) = \text{if } (a \in T) \text{ then } (T \cup \{b\}) \text{ else } T$. Argue that f is monotonic.

   How about the function $g(T) = (T - \{a\})$?

5. (10 points) If D is a domain and f:D→D and g:D→D are monotonic, show that the function h(x) = f(g(x)) is monotonic.

6. (10 points) Let D be a domain and let f:D→D and g:D→D be monotonic and extensive functions.

   (a) (4 points) Show that any solution to the following system of simultaneous equations:

   x = f(x)
   x = g(x)

   is a solution to the equation x = f(g(x)) and vice versa.

   (b) (2 points) From this and the result of Q(5), argue that this system of simultaneous equations always has a least solution, and describe how to compute it.

   (c) (2 points) Do the results of (a) and (b) always hold if f and g are monotonic but not extensive?

   (d) (2 points) Do the results of (a) and (b) always hold if f and g are extensive but not monotonic?

7. (10 points) Let $\Re$ denote the set of real numbers [0,1] ordered in the usual way by the $\leq$ relation. This is a totally ordered set whose least element is 0, and whose greatest element is 1. Consider a function f:$\Re \to \Re$.

   (a) (2 points) Suppose f is extensive. Give an informal description of what its graph may (and may not) look like, using a few pictures.

   (b) (2 points) Suppose f is monotonic. Give an informal description of what its graph may (and may not) look like, using a few pictures. Hint: the word ”derivative” might be useful here.

(c) (2 points) Complete the following sentence: the fixpoints of f, if they exist, must lie on the line _____.

(d) (2 points) Give an example of a function f:$\Re \rightarrow \Re$ without any fixpoints. Give an example of such a function that has one or more fixpoints.

(e) (2 points) Another famous fixpoint theorem is known humorously as the "hairy ball theorem." Read up about it and write a few sentences about this theorem.

(f) (Extra credit, 10 points)) Is monotonicity of f enough to guarantee that it has a least fixpoint? If so, give a proof. This is a difficult problem and its solution uses results that we did not cover in class.

8. (10 points) Answer the following questions briefly.

   - (2 points) What is the difference between a grammar and a language?
   - (4 points) What is an ambiguous grammar? Give an example of an ambiguous grammar and explain using an example why it is ambiguous.
   - (2 points) Can all context-free grammars be parsed using recursive-descent parsing? Explain using a simple grammar as a counter-example.
   - (2 points) What algorithm would you use to parse general context-free grammars? This was not covered in class so you will need to do some research on your own.

9. (20 points) SLL(1) grammars can be generalized in a natural way to SLL(k) grammars for which we use k lookahead symbols to make parsing decisions. The theory of SLL(k) parsers is based on two relations called $FIRST_k$ and $FOLLOW_k$ that generalize the $FIRST$ and $FOLLOW$ relations that we discussed in class. Some of the concepts used in the definitions below are defined at the end of this problem set. In the rest of this problem, assume that we are given a context-free grammar $G{=}<N, T, P, S>$ and that k is a fixed integer. The augmented grammar is $G'{=}<N', T', P', S'>$ (see definition below).

   - If α is a string of terminals and/or terminals, $FIRST_k(\alpha)$ is defined to be the set of k-prefixes of strings that can be produced from α by applying the productions of the grammar. One way to think about this is the following.

     If A is a non-terminal, $FIRST_k$(A) is defined as the set of k-prefixes of strings that can be derived from A using the grammar productions. The definition of $FIRST_k$ can be extended to strings of terminals and non-terminals as follows:

     $$FIRST_k(\epsilon) = \{\epsilon\},$$
     $$FIRST_k(t \in T) = \{t\},$$
     $$FIRST_k(u_1 u_2 ... u_n) = FIRST_k(u_1) +_k ... +_k FIRST_k(u_n).$$

     Let $\mathcal{M}$ be the finite lattice whose elements are sets of terminal strings of length at most k, ordered by containment with the empty set being the least

element. $FIRST_k$ sets for non-terminals can be computed as the least solution in $\mathcal{M}$ of this equational system:

$$\forall A \in N \;\; FIRST_k(A) = \bigcup_{A \to \alpha} FIRST_k(\alpha).$$

- $FOLLOW_k$ sets can be defined analogously. Let $\mathcal{L}$ be the lattice whose elements are sets of terminal strings of length exactly k for the augmented grammar, ordered by containment with the empty set being the least element. $FOLLOW_k$ sets for non-terminals other than S' can be computed as the least solution in $\mathcal{L}$ of this equational system:

$$FOLLOW_k(S) = \{\$^k\},$$
$$FOLLOW_k(B) = \bigcup_{A \to \alpha B \gamma} FIRST_k(\gamma) +_k FOLLOW_k(A),$$
$$\forall B \in (N' - \{S, S'\}).$$

i. (5 points) Explain briefly the intuition behind the equations for computing $FIRST_k$ and $FOLLOW_k$. How do these definitions avoid the need for computing NULLABLE non-terminals, as we did in class?

ii. (15 points) Consider the grammar

$$S \to yLab|yLbc|M,$$
$$L \to a|\epsilon,$$
$$M \to MM|x.$$

Write down the equational systems for the $FIRST_2$ and $FOLLOW_2$ sets for the non-terminals of the grammar. Solve these equations to compute these sets.

****

The following definitions are needed for the last problem.

- Given a string s of terminal symbols, the *k-prefix* of s, written as $(s)_k$, is the string consists of the first k symbols of s. If the length of the string is less than k, then the k-prefix is just the string itself. For example,

$$(abc)_2 = ab,$$
$$(abc)_1 = a,$$
$$(abc)_4 = abc.$$

- The operator $+_k$ takes two terminal strings and returns the k-prefix of their concatenation. This operator can be lifted to sets of strings in the obvious way. For example,

$$a +_2 bcd = ab,$$
$$a +_2 \epsilon = a,$$
$$\{\epsilon, t, tu, abc\} +_2 \{\epsilon, x, xy, xya\} = \{\epsilon, x, xy, t, tx, tu, ab\}.$$

- Lookahead computation is simplified if we pad the input string with k \$ symbols at the end; this ensures that we always have at least k symbols of lookahead even when we are near the end of the string. This can be described formally by defining an *augmented* grammar

  $$\text{G'=}<N'=N \cup \{S'\}, T'=T \cup \{\$\}, P'=P \cup \{S' \rightarrow S\$^k\}, S'>.$$

  Intuitively, we add a new non-terminal S', a new terminal symbol \$, and a production $S' \rightarrow S\$^k$ to the grammar, and make the new start symbol S'.