CS 380C: Problem set on Fixpoint Equations
Spring 2016

Due: February 11, 2016

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. 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) Draw a diagram of this partially ordered set. You do not have to show transitive edges.
   
   (b) Write down a function $D \to D$ that is monotonic but not extensive.
   
   (c) Write down a function $D \to D$ that is extensive but not monotonic.
   
   (d) Write down a function $D \to D$ that is both extensive and monotonic.
   
   (e) 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) Write down a function $f : D \to D$ for which the equation $x = f(x)$ has multiple solutions.

2. 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.

3. Let $D = (\mathcal{P}, \subseteq)$ be the domain consisting of the powerset of a set $S$ ordered by subset ordering, and let $a, b \in S$. Consider the function $f : \mathcal{P} \to \mathcal{P}$ 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 $g(T) = \text{if } (a \in T) \text{ then } (T - \{a\}) \text{ else } T$?

4. If $D$ is a domain and $f:D\to D$ and $g:D\to D$ are monotonic, show that the function $h(x) = f(g(x))$ is monotonic.

5. Let $D$ be a domain and let $f:D\to D$ and $g:D\to D$ be monotonic and extensive functions.
(a) 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) From this and the result of (2), argue that this system of simultaneous equations always has a least solution, and describe how to compute it.

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

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

6. In the fixpoint theorem we proved in class for domains \( D \) consisting of finite partially-ordered sets with a least element \( \bot \), we considered monotonic functions \( f \) and we considered an iteration of the form \( \bot, f(\bot), f(f(\bot)), \ldots \).

If \( a \) is an arbitrary element of the domain \( D \), and \( f \) is an arbitrary function (not necessarily monotonic), what is the strongest statement you can make about the sequence \( a, f(a), f(f(a)), \ldots \)? What can you say if \( f \) is monotonic?

7. 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 \( \text{FIRST}_k \) and \( \text{FOLLOW}_k \) that generalize the \( \text{FIRST} \) and \( \text{FOLLOW} \) 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 \( G = \langle N, T, P, S \rangle \) and that \( k \) is a fixed integer. The augmented grammar is \( G' = \langle N', T', P', S' \rangle \).

- If \( A \) is a non-terminal, \( \text{FIRST}_k(A) \) is defined as the set of \( k \)-prefixes of strings that can be derived from \( A \) using the grammar productions. The definition of \( \text{FIRST}_k \) can be extended to strings of terminals and non-terminals as follows.
  \[
  \text{FIRST}_k(\epsilon) = \{ \epsilon \} \\
  \text{FIRST}_k(t \in T) = \{ t \} \\
  \text{FIRST}_k(u_1 u_2 \ldots u_n) = \text{FIRST}_k(u_1) +_k \ldots +_k \text{FIRST}_k(u_n)
  \]

Let \( M \) 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. \( \text{FIRST}_k \) sets for non-terminals can be computed as the least solution in \( M \) of this equational system:
\[
\forall A \in N \quad \text{FIRST}_k(A) = \bigcup_{A \rightarrow \alpha} \text{FIRST}_k(\alpha)
\]

- \( \text{FOLLOW}_k \) sets can be defined analogously. Let \( L \) 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. \( \text{FOLLOW}_k \) sets for non-terminals other than \( S' \) can be computed as the least solution in \( L \) of this equational system:

\[
\text{FOLLOW}_k(S) = \{\text{\$}^k\}
\]

\[
\forall B \in (N' - \{S, S'\}). \text{FOLLOW}_k(B) = \bigcup_{A \rightarrow \alpha B \gamma} \text{FIRST}_k(\gamma) + k \text{FOLLOW}_k(A)
\]

i. Explain briefly the intuition behind the equations for computing \( \text{FIRST}_k \) and \( \text{FOLLOW}_k \). How do these definitions avoid the need for computing \( \text{NULLABLE} \) non-terminals, as we did in class?

ii. Consider the grammar

\[
S \rightarrow yLab | yLbc | M
L \rightarrow a | \varepsilon
M \rightarrow MM | x
\]

Write down the equational systems for the \( \text{FIRST}_2 \) and \( \text{FOLLOW}_2 \) 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,

\[
a +_2 bcd = ab
\]

\[
a +_2 \varepsilon = a
\]

\[
\{\varepsilon, t, tu, abc\} +_2 \{\varepsilon, x, xy, xya\} = \{\varepsilon, x, xy, t, tx, tu, ab\}
\]

- Lookahead computation is simplified if we pad the input string with \( k \) \( \text{\$} \) 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 \( G' = <N' = N \cup \{S'\}, T' = T \cup \{\text{\$}\}, P' = P \cup \{S' \rightarrow S \text{\$}^k\}, S' > \).

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