CS314: Principles of Programming Languages
Written Assignment 2

Nov. 28th, 2022

Name: Seifeldeen Mohamed.

NetID: Sm2632

- The written assignment has a total of **7** points. There is a total of 3 pages.
- For partial credit, show all of your work and clearly indicate your answers.
- You can either annotate your solution on this document or put your solution in another text document (e.g. MS Word) with clear marks to label the answer to each question.
- Submit a PDF version of your solution to Canvas (e.g. using the printing function of Word.)

Prolog

1. **(2 points)** True or False

   (a) **($\frac{1}{2}$ point)** In Prolog, A+b unifies with b+A. (T)/F

   (b) **($\frac{1}{2}$ point)** Reordering the terms in the body of a Prolog rule may change the result. (T)/F

   (c) **($\frac{1}{2}$ point)** The result of the query **?- 3 is A + 1.** is A = 2. T/(F)

   (d) **($\frac{1}{2}$ point)** With occurs_check enabled, a Prolog query can avoid infinite search. (T)/F

2. **(2 points)** What is the unifier of each of the following terms? Assume that occurs_check is true.

   (a) **($\frac{1}{2}$ point)** f(X,Y,Z) = f(Y,Z,X)

   A. {X/Y, Y/Z}
   B. {X/Y, Z/y}
   (C) {X/A, Y/A, Z/A}
   D. None of the above.

   • terms does not satisfy occurs-check

   (b) **($\frac{1}{2}$ point)** tree(X,tree(X,a)) = tree(Y,Z)

   A. Does not unify.
   B. {X/Y, Z/tree(X,a)}
   (C) {X/Y, Z/tree(Y,a)}
   D. {Y/X, Z/tree(Y,a)}

   • The unifier of these two terms is the substitution of X for Y and Z for tree(X,a). This satisfies the occurs_check, which means no variables can be substituted with another variable containing the same variable

   (c) **(1 point)** [A,B,C] = [(B,C),b,a(A)]

   A. Does not unify.
   B. {A/(b,a(A)), B/b, C/a(A)}
   C. {A/(b,a(C)), B/b, C/a(A)}
   (D) None of the above.

   • A replaced with (b, a(A)), B with b and C with a(A). satisfies the occur_check

3. **(2 points)** Fill in the implementation of segment(A,B) predicate below, which holds when A is a contiguous segment contained anywhere within list B. You may use prefix, suffix and append. Do not provide code for these functions. For example:

```
?- segment ([3,5], [1,2,3,4,5]).
false .

?- segment ([X,Y], [1,2,3,4]).
X = 1, Y = 2;
X = 2, Y = 3;
X = 3, Y = 4;
false .

?- segment ([3,4,X], [1,2,3,4,5]).
X = 5;
false .
```

× Prefix precedes its single argument
× append(X, B, A) is [..., X, ... B] == A: the list of elements of X, followed by elements of B, together, is in the list of elements in A.

```
append([], A, A).
append([Left|Rist], L2, [Left|L3]) :- append(Rist, L2, L3)
prefix(Pre, L) :- append(Pre, _, L)
suffix(Suff, L) :- append(_, Suff, L).
segment(Seg, L, L) :- suffix(S, L), prefix(Seg, S).
```

4. **(1 point)** In this problem we will write a matrix transpose function in Python. A matrix is a two-dimensional array, which we will represent as a list of lists of integers. For example, the following is a 2×3 matrix:

```
A = [[1, 2, 3],
     [4, 5, 6]]
```

The transpose of a matrix A of dimension $n \times m$ is a matrix B of dimensions $m \times n$ such that A[i][j] is equal to B[j][i], for all valid indices i and j into matrix A. For example:

```
>>> transpose ([[1, 2, 3],
                [4, 5, 6]])
  [[1, 4],
   [2, 5],
   [3, 6]]
```

Your code must be in this form:

```
def transpose(m):
    height = len(m)
    width = len(m[0])
    return [ [ ------ for ----- in ------] for ---- in ------]
```

Fill in the return statement of transpose below:

```
return [[m[j][i] for j in range (height)] for i in range
    (width)]
```