

# CS 560: Homework 2

Eric Stevens

October 14, 2018

## Question 1:

To solve this problem lets look at the truth table for the expression:

$p(X)$	$q(X)$	$p(X) \leftarrow q(X)$
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	FALSE
FALSE	FALSE	TRUE

This thruth table reflect the larger issue at hand. We can see that 3 of the 4 expressions evaluate to TRUE meaning that for 3 out of 4 interpretations of X that statment will be a bodel.

Phrased in another way, if the following statement was in KB:

$$p(a) \leftarrow q(a)$$

then we would be able to say that 3/4 of the total number of interpetations are models of the knowledgebase. The fact that there is a variable in the expression adds complexity. We need to account for every constant in each interpretation. Since each one of the three constants needs to force this rule, we can multiply the total number of interpetstions by  $(3/4)^3$ . The number of interpretations that are models of this knowledge is calculated as follows:

$$total\ interpretations * \left( \frac{truth\ table\ trues}{truth\ table\ totals} \right)^{constants}$$

$$16,384 * \left( \frac{3}{4} \right)^3$$

$$6,912\ Models$$

## Question 1 Critique:

My approach differed from the answer sheet in that mine was a top-down route, starting with the known number of total interpretations. In this question I failed to remember that it is the number of individuals in the domain that are important in terms of the calculation, rather than the constants,

since the constants can map to all of the domain individuals. Instead of making the exponent of the the above function the number of constants it should have been the number of individuals in the domain. This is demonstrated as follows:

$$\begin{aligned} & \text{total interpretations} * \left( \frac{\text{truth table trues}}{\text{truth table totals}} \right)^{\text{Domain}} \\ & 16,384 * \left( \frac{3}{4} \right)^4 \\ & 5,184 \text{ Models} \end{aligned}$$

## Question 2:

"You can do a bottom-up proof procedure for the cause with variables by carrying out the bottom-up procedure on the set of all ground instances of the clauses. A ground instance of a clause is obtained by uniformly substituting constants appearing in the knowledge base or in any query for the variables in the clauses." (Cl. p.54) In this case the variable  $a$  doesn't appear in any clause or query in the knowledge base, but we have been explicitly told that  $a$  is a constant in our syntax. Therefore, the proof procedure is valid.

We have the knowledge base:  $KB = \{p(X)\}$

For our bottom-up proof we take the clauses in our knowledge base and uniformly substitute our constants in for the variables in the clauses. In this case we have only one clause in our knowledge base ( $p(X)$ ) and only one known constant in our syntax ( $a$ ).

The ground instance we obtain from the bottom-up proof procedure is:

yes  $\leftarrow p(X)$

- **Sub:**{X/a}

yes  $\leftarrow p(a)$

Therefore it can be said that  $KB$  **implies**  $p(a)$

Furthermore we can say that because we must substitute constants for variables uniformly, any constant that is to exist in our syntax will have a similar resulting ground instance. The introduction of a statement like  $p(X)$  into a knowledge base will double the number of interpretations but do nothing to the number of models.

## Question 2 Critique

I did not understand the grammar of the question and, as a result, did not perform the type of formal proof in the way the answer sheet does. When I saw 'proof' my mind went to top-down or bottom-up. I first tried top-down and failed to make sense of the problem. When I saw the definition of bottom-up, I felt that it was sufficient for solving this problem.

I also attempted to describe the implications of the proof in terms of the of the knowledge base, implementations, models, and variable assignments in order to fulfill the requirements of the question. This was done in paragraph form rather than formal proof form.

### Question 3:

?p(e)

yes  $\leftarrow$  p(e)

- **Use:**  $p(X1) \leftarrow q(X1) \wedge r(X1)$
- **Sub:** { X1/e }

yes  $\leftarrow$  q(e)  $\wedge$  r(e)

- **Use:** r(e)
- **Sub:** { }

yes  $\leftarrow$  q(e)

- **Use:**  $q(X2) \leftarrow s(X2, Y2) \wedge q(Y2)$
- **Sub:** { X2/e, Y2,d }

yes  $\leftarrow$  s(e, d)  $\wedge$  q(d)

- **Use:** s(e, d)
- **Sub:** { }

yes  $\leftarrow$  q(d)

- **Use:**  $q(X3) \leftarrow s(X3, Y3) \wedge q(Y3)$
- **Sub:** { X3/d, Y3/b }

yes  $\leftarrow$  s(d, b)  $\wedge$  q(b)

- **Use:** s(d, b)
- **Sub:** { }

yes  $\leftarrow$  q(b)

- **Use:** q(b)
- **Sub:** { }

yes  $\leftarrow$

Since we end up with a yes in the head and an empty body, we can conclude that KB implies p(e).

### Question 3 Critique:

This proof is correct. You may notice that there were substitutions that look different from the answer sheet. Some substitutions were made in advance of the usage of the clause that would resolve them. This would not be the way that an algorithm would perform a substitution, but the

problem stated "you are free to pick which conjunct in the answer clause you want to work on", which made me think that we did not need to strictly adhere to an algorithmic process. It seems that this type of early substitution will not cause errors later.

## Question 4:

?p(e)

yes  $\leftarrow$  p(e)

- **Use:**  $p(X1) \leftarrow q(X1) \wedge r(X1)$
- **Sub:** { X1/e }

yes  $\leftarrow$  q(e)  $\wedge$  r(e)

- **Use:** r(e)
- **Sub:** { }

yes  $\leftarrow$  q(e)

- **Use:**  $q(X2) \leftarrow s(X2, Y2) \wedge q(Y2)$
- **Sub:** { X2/e, Y2/f } THIS IS WHERE f IS USED

yes  $\leftarrow$  s(e, f)  $\wedge$  q(f)

- **Use:** q(f)
- **Sub:** { }

yes  $\leftarrow$  s(e, f)

- **FAIL**

Here the derivation fails to come to a conclusion. This failure does not imply that it is not possible to derive p(e), it simply implies that there was a conjunct choice or substitution choice that led down a derivation path that could not resolve the query. In fact, we could have performed the same derivation as we did in the previous question; the introduction of p(f) would not change anything.

## Question 5:

? s(e, Y)  $\wedge$  q(Y)

yes  $\leftarrow$  s(e, Y)  $\wedge$  q(Y)

- **Use:**  $q(X) \leftarrow s(X, Y) \wedge q(Y)$
- **Sub:** { X/Y }

yes  $\leftarrow$  s(e, Y)  $\wedge$  s(Y, Y)  $\wedge$  q(Y)

We could work this problem but its failure to resolve is already obvious. When the clause was chosen to replace the q(Y) atom the variables were not refreshed. Therefore, when the substitution was made, the new variables that were introduced shared a name with some of the old variables.

This reduced the flexibility of the future substitutions. With the introduction of the atom  $q(Y,Y)$ , which our poor substitution forced, we immediately know that we will not be able to resolve this query. We come to this realization through an inspection of the knowledge base. In KB we can see that there is no fact that allows the  $s$  predicate to contain two of the of the same constants. We also see that there is no clause with the  $s$  predicate in the head, so there is no way to add further  $s$  ground instances.

The way we should have performed the substitution is by refreshing the variables in the clause we choose to replace the  $q(Y)$  atom in the query. The proper substitution and the working of the problem follows:

$yes \leftarrow s(e, Y) \wedge q(Y)$

- **Use:**  $q(W) \leftarrow s(W, Z) \wedge q(Z)$  this is where we refresh the variables
- **Sub:**  $\{ W/Y \}$

$yes \leftarrow s(e, Y) \wedge s(Y, Z) \wedge q(Z)$

- **Resolve with:**  $s(e, d)$
- **Sub:**  $\{ Y/d \}$

$yes \leftarrow s(d, Z) \wedge q(Z)$

- **Resolve with:**  $s(d, b)$
- **Sub:**  $\{ Z/b \}$

$yes \leftarrow q(b)$

- **Resolve with:**  $q(b)$
- **Sub:**  $\{ \}$

$yes \leftarrow$

As we can see from the above top-down proof, a solution can be obtained by execution proper substitution.

## Question 6:

### Part (a)

*Substitution of  $f(A, X, Y, X, Y)$   
with  $A/X, Z/b, Y/c$   
 $f(X, X, c, X, c)$*

### Part (b)

*Substitution of  $yes(F, L) \leftarrow append(F, c(L, nil), c(l, c(i, c(s, c(t, nil))))))$   
with  $F/c(l, X1), Y1/c(L, nil), A1/l, Z1/c(i, c(s, c(t, nil)))$   
 $yes(c(l, X1), L) \leftarrow append(c(l, X1), c(L, nil), c(l, c(i, c(s, c(t, nil))))))$*

### Part (c)

*Substitution of  $\text{append}(c(A1, X1), Y1, c(A1, Z1)) \leftarrow \text{append}(X1, Y1, Z1)$*   
*with  $F/c(l, X1), Y1/c(L, \text{nil}), A1/l, Z1/c(i, c(s, c(t, \text{nil})))$*   
 *$\text{append}(c(l, X1), c(L, \text{nil}), c(l, c(i, c(s, c(t, \text{nil}))))$*   
 *$\leftarrow \text{append}(X1, c(L, \text{nil}), c(i, c(s, c(t, \text{nil}))))$*

## Question 7:

**Part (a):**

$\{Z/f(X), Y/g(b)\}$

**Part (b):**

$\{X/t, W/f(t), Q/t\}$

**Part (c):**

$\{Z/\text{val}(X, \text{bb}), P/\text{val}(X, \text{bb})\}$

## Question 8:

**Rule:**  $q(Y) \leftarrow s(Y, Z) \wedge r(Z)$

**Atoms:**  $s(f(a), b), r(b)$

**Sub List:**  $\{Y/f(a), Z/b\}$

**Add to C:**  $q(f(a))$

**Rule:**  $q(Y) \leftarrow s(Y, Z) \wedge r(Z)$

**Atoms:**  $s(f(b), b), r(b)$

**Sub List:**  $\{Y/f(b), Z/b\}$

**Add to C:**  $q(f(b))$

**Rule:**  $q(Y) \leftarrow s(Y, Z) \wedge r(Z)$

**Atoms:**  $s(c, b), r(b)$

**Sub List:**  $\{Y/c, Z/b\}$

**Add to C:**  $q(c)$

## Question 8 Critique:

I made major mistakes in this problem. My first mistake was a simple one. I did not realize that the clauses in the knowledge bases had to be added to C. I thought that by definition the knowledge base was already a part of C.

Then, I failed to keep track of an up to date consequence list, therefore assuming that the only derivations that can be made are those that can be derived from the clauses in the knowledge base.

To be candid, I clearly did not understand this problem. I will rework this problem below for my own understanding:

**Rule:**  $s(f(a), b)$

**Atoms:**  $s(f(a), b)$

**Sub List:**  $\{ \}$

**Add to C:**  $\{ s(f(a), b) \}$

**Rule:**  $s(f(b), b)$

**Atoms:**  $s(f(b), b)$

**Sub List:**  $\{ \}$

**Add to C:**  $\{ s(f(a), b), s(f(b), b) \}$

**Rule:**  $s(c, b)$

**Atoms:**  $s(c, b)$

**Sub List:**  $\{ \}$

**Add to C:**  $\{ s(f(a), b), s(f(b), b), s(c, b) \}$

**Rule:**  $r(b)$

**Atoms:**  $r(b)$

**Sub List:**  $\{ \}$

**Add to C:**  $\{ s(f(a), b), s(f(b), b), s(c, b), r(b) \}$

**Rule:**  $q(Y) \leftarrow s(Y, Z) \wedge r(Z)$

**Atoms:**  $s(f(a), b), r(b)$

**Sub List:**  $\{ Y/f(a), Z/b \}$

**Add to C:**  $\{ s(f(a), b), s(f(b), b), s(c, b), r(b), q(f(a)) \}$

**Rule:**  $q(Y) \leftarrow s(Y, Z) \wedge r(Z)$

**Atoms:**  $s(f(b), b), r(b)$

**Sub List:**  $\{ Y/f(b), Z/b \}$

**Add to C:**  $\{ s(f(a), b), s(f(b), b), s(c, b), r(b), q(f(a)), q(f(b)) \}$

**Rule:**  $q(Y) \leftarrow s(Y, Z) \wedge r(Z)$

**Atoms:**  $s(c, b), r(b)$

**Sub List:**  $\{ Y/c, Z/b \}$

**Add to C:**  $\{ s(f(a), b), s(f(b), b), s(c, b), r(b), q(f(a)), q(f(b)), q(c) \}$

**Rule:**  $p(X) \leftarrow q(f(X))$

**Atoms:**  $q(f(a))$

**Sub List:**  $\{ X/a \}$

**Add to C:**  $\{ s(f(a), b), s(f(b), b), s(c, b), r(b), q(f(a)), q(f(b)), q(c), p(a) \}$

**Rule:**  $p(X) \leftarrow q(f(X))$

**Atoms:**  $q(f(b))$

**Sub List:**  $\{ X/b \}$

**Add to C:**  $\{ s(f(a), b), s(f(b), b), s(c, b), r(b), q(f(a)), q(f(b)), q(c), p(a), p(b) \}$

## Question 9:

KB = {

dog(fido)

love(john, mary)

```

love(john, X) ← dog(X)

faster(X, Y) ← horse(X) ^ dog(Y)

faster(X, fido) ← horse(X)
}

```

## Question 9 Critique:

I believe my answer to this is correct. Although I do have the extra line "faster(X, fido) ← horse(X)" I believe that this line could be seen as part of the datalog interpretation.

## Question 10:

```

In [8]: def substitute(expr,subs):

    # declare the substituted list
    newexpr = [expr[0]]

    # loop through input arguments
    for arg in expr[1:]:

        # if argument exist in substitution list then sub it
        if arg in subs: newexpr.append(subs[arg])

        # if the argument is not in the sub list then let it pass through
        else: newexpr.append(arg)

    return newexpr

def test(atom,subs):
    result = substitute(atom,subs)
    str = "Substitution of %s with" % prettyexpr(atom)
    for v in subs:
        str += " %s/%s" % (v,prettyexpr(subs[v]))
    print("%s\n %s" % (str,prettyexpr(result)))

def doall():
    # here is a variant of Exercise 2.9 part a)
    test(["fun", "A", "X", "Y", "X", "Y"], {"A" : "X", "Z" : "b", "Y" : "c"})

doall()

```

Substitution of fun(A,X,Y,X,Y) with A/X Z/b Y/c  
 fun(X,X,c,X,c)

## Question 11:



I have elected to use a slightly different strategy for the 'substitute()' than in the provided code. The 'test()' function is also heavily modified as required by the assignment. The prettyexpr is exactly the same. The altered code is heavily commented for ease of understanding.

```

In [7]: def substitute(expr,subs):

    # deal with single variable
    if type(expr[0]) is str:
        if expr[0][0].isupper(): return [subs[expr[0]]]
        # this is hacky, I am doing this because this is what the assignment
        # requests but I think that the program should fail here. From what
        # I know of Datalog, terms cannot exist outside of predicates. I th
        # this situation should return an 'invalid expression' warning.

    # list to hold the substituted output string
    newexpr = [expr[0]]

    # loop through each areument in input list
    for arg in expr[1:]:

        # If item is a list, recursivly call 'substitute()'
        if type(arg) is list: newexpr.append(substitute(arg,subs))

        # if the argument is in the sub list then sub it
        elif arg in subs: newexpr.append(subs[arg])

        # otherwise, let it pass through
        else: newexpr.append(arg)

    return newexpr

def test(atom,subs):
    result = substitute(atom,subs)
    str = "Substitution of %s with" % prettyexpr(atom)
    for v in subs:
        str += " %s/%s" % (v,prettyexpr(subs[v]))
    print("%s\n %s" % (str,prettyexpr(result)))

def doall():

    # set part b attributes
    b_head = ["yes", "F", "L"]
    b_body = ["append", "F", ["c", "L", "nil"], ["c", "l", ["c", "i", ["c",
    b_subs = {"F":["c", "l", "X1"], "Y1":["c","L","nil"], "A1":"l", "Z1":["c

    # set part c attributes
    c_head = ["append",["c", "A1", "X1"], "Y1", ["c", "A1", "Z1"]]
    c_body = ["append", "X1", "Y1", "Z1"]
    c_subs = {"F":["c", "l", "X1"], "Y1":["c", "L", "nil"], "A1":"l", "Z1":["c

    # Run test on head of part b
    print("\n\n(b) head:")
    test(b_head, b_subs)

    # Run test on head of part b
    print("\n\n(b) body:")
    test(b_body, b_subs)

    # Run test on head of part b
    print("\n\n(c) head:")

```

```

test(c_head, c_subs)

# Run test on head of part b
print("\n\n(c) body:")
test(c_body, c_subs)

# Run test on head of part b
print("\n\nSingle Variable:")
test(["X"], {"X": "Y"})

doall()

```

(b) head:

Substitution of yes(F,L) with F/c(l,X1) Y1/c(L,nil) A1/l Z1/c(i,c(s,c(t,nil)))  
 yes(c(l,X1),L)

(b) body:

Substitution of append(F,c(L,nil),c(l,c(i,c(s,c(t,nil)))) with F/c(l,X1) Y1/c(L,nil) A1/l Z1/c(i,c(s,c(t,nil)))  
 append(c(l,X1),c(L,nil),c(l,c(i,c(s,c(t,nil))))

(c) head:

Substitution of append(c(A1,X1),Y1,c(A1,Z1)) with F/c(l,X1) Y1/c(L,nil) A1/l Z1/c(i,c(s,c(t,nil)))  
 append(c(l,X1),c(L,nil),c(l,c(i,c(s,c(t,nil))))

(c) body:

Substitution of append(X1,Y1,Z1) with F/c(l,X1) Y1/c(L,nil) A1/l Z1/c(i,c(s,c(t,nil)))  
 append(X1,c(L,nil),c(i,c(s,c(t,nil))))

Single Variable:

Substitution of X with X/Y  
 Y