# CS322 Fall 1999
# Module 3 (Reasoning with Symbols)
# Assignment 3 — Solution

The aim of this assignment is to learn about proof procedures, and how to compute logical consequences.

## Question 1

The file plumbing2.pl (available on the web and in `~cs322/cilog/`) contains a CILog axiomatization for the solution to assignment 1.

(a) Give a sequence of atoms added to the consequence set for a bottom-up proof procedure. Show clearly what is the consequence set.

(b) Give a top-down derivation for the query

?*flow*(*d*1).

(c) Give a failing derivation for the query ?*flow*(*d*1).

### Solution

(a) Give a sequence of atoms added to the consequence set for a bottom-up proof procedure. Show clearly what is the consequence set.

Here is one such sequence:

```
pressurised(p1)
on(t1)
pressurised(p2)
pressurised(p3)
on(t2)
flow(shower)
wet(bath)
off(t3)
off(t4)
off(t5)
on(t6)
unplugged(bath)
flow(d2)
flow(d1)
plugged(sink)
```

This is the sequence that results from selecting the topmost applicable rule at each stage. Many other sequences are possible, as long as the body of a clause is derived before the head.

(b) Give a top-down derivation for the query

?*flow*(*d*1).

Here is such a derivation (always selecting the leftmost atom in the body):

```
yes <- flow(d1).
yes <- flow(d2).
yes <- wet(bath) & unplugged(bath).
yes <-  flow(shower) & unplugged(bath).
yes <- on(t2) & pressurised(p2) & unplugged(bath).
yes <- pressurised(p2) & unplugged(bath).
yes <- on(t1) & pressurised(p1) & unplugged(bath).
yes <- pressurised(p1) & unplugged(bath).
yes <- unplugged(bath).
yes <- .
```

(c) Give a failing derivation for the query ?*flow*(*d*1).

Here is a failing derivation (always selecting the leftmost atom in the body):

```
yes <- flow(d1).
yes <- flow(d3).
yes <- wet(sink) & unplugged(sink).
yes <- on(t3) & pressurised(p3) & unplugged(sink).
```

This fails as there is no clause with *on*(*t*3) in the head.

## Question 2

Consider the following Datalog program (also available here) with clauses numbered:

1 : $attends(P, D) \leftarrow proctors\_exam(P, C) \wedge exam(C, D)$.

2 : $proctors\_exam(P, C) \leftarrow instructor(P, C)$.

3 : $proctors\_exam(P, C) \leftarrow teaching\_assistant(P, C)$.

4 : $exam(cs322, oct22)$.

5 : $exam(cs322, dec24)$.

6 : $instructor(david, cs322)$.

7 : $teaching\_assistant(rita, cs322)$.

8 : $teaching\_assistant(leslie, cs322)$.

CILog always selects the leftmost conjunct to resolve, and always chooses the topmost clause that leads to a solution.

(a) Give a top-down derivation, showing all substitutions, for the first answer for the query:

?*attends*(*W*, *T*).

Specify which clause was chosen at each stage.

(b) Give a top-down derivation, showing all substitutions, for the second answer for the same query. Only show the part of the derivations that is not shared with the first answer.

## Solution

(a) Give a top-down derivation, showing all substitutions, for the first answer for the query:

?*attends*(*W*, *T*).

Specify which clause was chosen at each stage.

Here is the derivation, always selecting the leftmost atom in the body:

```
yes(W,T) <- attends(W,T).
  choose clause 1, (with variables renamed) substitution: W/P1,T/D1
yes(P1,D1) <- proctors_exam(P1,C1) & exam(C1,D1).
   choose clause 2, substitution  P1/P2,C1/C2
yes(P2,D1) <- instructor(P2,C2) & exam(C2,D1).
   choose clause 6, substitution  P2/david, C2/cs322
yes(david,D1) <- exam(cs322,D1)
   *** choose clause 4, substitution D1/oct22
yes(david,oct22) <-
```

You solution should be the same, up to renaming of variables.

(b) Give a top-down derivation, showing all substitutions, for the second answer for the same query. Only show the part of the derivations that is not shared with the first answer.

```
   at *** choose clause 5, substitution D1/dec24
yes(david,dec24) <-
```