**CS550 \ Assignment 7**

Group 1

Ariel Stolerman

Bekah Overdorf

Sam Snodgrass

**4.59)**

a.

(meeting ?x (Friday ?y))

b.

(rule (meeting-time ?person ?day-and-time)

(or (meeting whole-company ?day-and-time)

(and (meeting ?m ?day-and-time)

(job ?person (?m . ?x)))))

c.

(meeting-time (Hacker Alyssa P) (Wednesday ?x))

**4.65)**

The wheel rule applies to employees that are at least “second-level” managers, i.e. supervisors of supervisors. Since wheel returns an entry for any distinct sequence of <employee, supervisor, supervisor-of-supervisor>, and Oliver appears in 4 such sequences, he appears 4 times in the query (wheel ?who). The sequences in which Oliver appears are:

<(Hacker Alyssa P), (Bitdiddle Ben), (Warbucks Oliver)>

<(Fect Cy D), (Bitdiddle Ben), (Warbucks Oliver)>

<(Tweakit Lem E), (Bitdiddle Ben), (Warbucks Oliver)>

<(Cratchet Robert), (Scrooge Eben), (Warbucks Oliver)>

**4.68)**

Scheme:

(rule (reverse () ()))

(rule (reverse (?u . ?v) ?y)

(and (reverse ?v ?r)

(append-to-form ?r (?u . ()) ?y)))

The reverse rules above work on queries like (reverse (1 2 3) ?x) but not on queries of the form (reverse ?x (1 2 3)).

We think it is because when the first argument is given a variable, like ?x, it can never reach the base case rule of (reverse () ()), because it doesn’t have a finite defined list to work with.

Prolog:

app([],Y,Y).

app([U|V],Y,[U|Z]) :- app(V,Y,Z).

rev([],[]).

rev([U|V],Y) :- rev(V,R), app(R,[U],Y).

In Prolog it works both ways (almost the same):

| ?- rev([1,2,3],U).

U = [3,2,1]

yes

| ?- rev(U,[1,2,3]).

U = [3,2,1] ? // must press <enter>

yes

**4.75)**

(define (unique-query exps) (car exps))

;; uniquely-asserted

;; works similar to negate, but takes only streams of length 1 (instead of null streams)

(define (uniquely-asserted operands frame-stream)

(stream-flatmap

(lambda (frame)

(let ((qevaled (qeval (unique-query operands)

(singleton-stream frame))))

(if (eq? (stream-length qevaled) 1)

qevaled

the-empty-stream)))

frame-stream))

;; added to initialize-data-base:

;; (put 'unique 'qeval uniquely-asserted)

A query that lists all people who supervise precisely one person:

(and (supervisor ?x ?y) (unique (supervisor ?z ?y)))