CS 240A : Databases and Knowledge Base
Homework #1

Ronak Sumbaly
UID : 604591897

October 7, 2015

Question 8.1

Query : To find those suppliers who supply all basic parts.

% Get all Suppliers who supply all basic parts
> suppliersAllBasicPart(Supplier) $\leftarrow$ part_cost(_, Supplier, _ , _), $\sim$ missingPart(Supplier).

% To check whether a specific supplier has all the basic parts
> missingPart(Supplier) $\leftarrow$ part_cost(Part$_a$, _, _), $\sim$ hasPart(Supplier, Part$_a$).

% Return table of Suppliers having a certain Part$_a$
> hasPart(Supplier,Part$_a$) $\leftarrow$ part_cost(Part$_a$, Supplier, _, _).

Question 8.2

Query : To find students who have taken at least two classes, and got the highest grade (possibly with others) in every class they took.

% Get all students with atleast two classes and highest grade in those classes.
> studentHighestGrade(Name) $\leftarrow$ student(Name, _ , _), atleastTwoClasses(Name), $\sim$ higherGrade(Name)

% Get all students who have taken atleast two classes.
> atleastTwoClasses(Name) $\leftarrow$ took(Name,Course$_1$, _), took(Name,Course$_2$, _), Course$_1$ $\sim=$ Course$_2$

% Get all student who have the highest grade in all his/her classes
> higherGrade(Name) $\leftarrow$ took(Name$_1$,Course, Grade$_1$), took(Name$_2$,Course, Grade$_2$), Grade$_1$ $<$ Grade$_2$

Question 8.5

(a.) **Prove**

Show that the expressive power of RA does not change if we drop set intersection.

Proof

Expressive power of relational algebra is the scope of all the outputs that can be constructed using the language.

In order to prove that the expressive power of relational algebra doesn't change if we drop the set intersection operator we need to prove that the latter can be implemented using some other operator present in the relational algebra so that all the outputs initially constructed using the language remain intact. We can show that in two following ways,

1. Set intersection can be constructed by taking an Equi - Join $\bowtie$ of the two relations for every column and then projecting the columns that are duplicated in the output
2. Set intersection can be constructed using the following equation: $A \cap B = A - (A - B) = B - (A - B)$

Hence we can conclude that the expressive power of relational algebra does not change if we drop set intersection.

(b.) **List of Monotonic Operators of RA**

List of Monotonic Operator	
Symbol	Operator
σ	Selection
π	Projection
$\cup$	Union
$\times$	Cartesian Product
$\bowtie$	Join

(c.) **Prove**

Show that the if we drop set difference then there is loss in expressive power of RA

Proof

Our claim that the expressive power of relational algebra doesn't change if we can prove that the operator being dropped can be constructed using some other operator present in relational algebra doesn't hold true for set difference. Since set difference is a basic operator it cannot be expressed as any combination of any other operator in RA.

Hence we can conclude that there is loss in expressive power of RA if we drop set difference.

Question 8.7

Rules and Predicate

$$r_1 : p(X,X) \leftarrow b2(Y,Y,a), b1(X), X > Y \tag{1}$$

$$r_2 : q(X,Y) \leftarrow p(X,Z), p(Z,Y) \tag{2}$$

$$r_3 : s(X) \leftarrow b2(Y,Y,a), X > Y, \neg b1(X) \tag{3}$$

Rule 1. The rule comprises of variables X, Y and constant a. Since the variables are contained in a positive base predicate/goal $b1$ and $b2$ the variables are $safe$. Since the body of the rule is $safe$ we can conclude that the the entire rule $p(X,X)$ is $SAFE$.

Relational Algebra Expression:

Step 1. Since the expression does not contain any equality sub-expression, we continue on to the next step.

Step 2. Translate the body of the expression into relational algebra.

$$Body_r = \sigma_{\$_1=\$_2,\$_3=a,\$_4>\$_1}(b_2 \times b_1) \tag{4}$$

where, $\$_1 = Y, \$_2 = Y, \$_3 = a, \$_4 = X$

Step 3. Translate each rule into a generalized projection on $Body_r$, according to patterns in the head of r.

$$S = \pi_{\$_4,\$_4} Body_r \tag{5}$$

Rule 2. The rule comprises of variables X, Y, Z . Since the variables are contained in a predicate p which is proved to be safe in r_1 we can say that the variables are $safe$. Since the body of the rule is $safe$ we can conclude that the the entire rule $q(X,Y)$ is $SAFE$.

Relational Algebra Expression:

Step 1. Since the expression does not contain any equality sub-expression, we continue on to the next step.

Step 2. Translate the body of the expression into relational algebra.

$$Body_r = \sigma_{\$_2=\$_3}(p \times p) \tag{6}$$

where, $\$_1 = X, \$_2 = Z, \$_3 = Z, \$_4 = Y$

Step 3. Translate each rule into a generalized projection on $Body_r$, according to patterns in the head of r.

$$S = \pi_{\$_1,\$_4} Body_r \tag{7}$$

Rule 3. The rule comprises of variables X, Y and constant a. Since the variable X is contained in a negative goal $\neg b1(X)$ we directly conclude that the rule r_3 is *NOT SAFE*.

Datalog Queries

example1.fac

```
%Facts

% city(Name:string, State:string, Population:integer)
city('Houston', 'Texas', 3000000).
city('Dallas', 'Texas', 2000000).
city('Huntsville', 'Texas', 150000).
city('Austin', 'Texas', 750000).
city('Corsicana', 'Texas', 60000).
city('Shreveport', 'Louisiana', 90000).
city('Bastrop', 'Texas', 6000).
city('San Antonio', 'Texas', 1500000).

% distance(City1:string, City2:string, Distance:float)
distance('Houston', 'Bastrop', 130.0).
distance('Houston', 'Huntsville', 60.0).
distance('Huntsville', 'Dallas', 100.0).
distance('Austin', 'Waco', 110.0).
distance('Waco', 'Dallas', 100.0).
distance('Dallas', 'Shreveport', 200.0).
distance('Austin', 'Bastrop', 30.0).
distance('Austin', 'San Antonio', 80.0).
distance('San Antonio', 'Houston', 190.0).
```

example1.deal

```
% Schema

database(city(Name:string, State:string, Population:integer), distance(City1:string, City2:string,
Distance:float) ).

% DeAL Queries

% Q1. Write an DeAL program to find the cities which are reachable from Austin, with their distance
% from Austin.

fromAustin(City,Distance) <- distance('Austin',City,Distance).

% Q2. Find which of the above city is the furthest from Austin.
farthestAustin(City,Distance) <- fromAustin(City,Distance), ~ fartherCity(City,Distance).
fartherCity(City2,Distance2) <- distance('Austin', City1, Distance1), distance('Austin', City2,
Distance2), Distance2 < Distance1.

% Q3. Write an DeAL program to add up the population of cities in Texas.
popTexas(sum<Popu>) <- city( _ , 'Texas', Popu)

export fromAustin(X,Y).
export farthestAustin(X,Y).
export popTexas(Y).
```

Query 1

Write an DeAL program to find the cities which are reachable from Austin, with their distance from Austin.

```
> fromAustin(City,Distance) ← distance('Austin',City,Distance).
```

Query 2

Find which of the above city is the furthest from Austin.

```
> farthestAustin(City,Distance) ← fromAustin(City,Distance), ~ fartherCity(City,Distance).
> fartherCity(City2,Distance2) ← distance('Austin', City1, Distance1), distance('Austin', City2,
Distance2), Distance2 < Distance1.
```

Query 3

Write an DeAL program to add up the population of cities in Texas.

```
> popTexas(sum<Popu>) ← city( _ , 'Texas', Popu)
```