CS240A Homework 2

Author: Sneha Shankar

UID: 404946026

a) Rewrite the fastest query in Example 8.17 by using the aggregate min

fastest(Part, min<Time>) <- part_cost(Part, Sup, Cost, Time).

b) Rewrite the howsoon query in Exampe 8.19 as an exo-max program

fastest(Part, min<Time>) <- part_cost(Part, Sup, Cost, Time).

timeForBasic(AP, Time) <- fastest(AP,Time).
timeForBasic(AssPart, Time) <- assembly(AssPart, BasicSub,_), timeForBasic(BasicSub, Time).

howsoon(AssPart, max<Time>) <- timeForBasic(AssPart, Time).

c) Rewrite query (b) as endo -max program

fastest(Part, min<Time>) <- part_cost(Part, Sup, Cost, Time). .....(i)

timeForBasic(AP, max<Time>) <- fastest(AP,Time). .....(ii)

timeForBasic(AssPart, max<Time>) <- assembly(AssPart, BasicSub,_), timeForBasic(BasicSub, Time). .....(iii)

howsoon(P,T) <- timeForBasic(P,T). .....(iv)

d) Explain why (c) has the PreM property

In question (c) above, recursion is defined by (iii) and it's exit rule is defined by (ii). We know that PreM always holds for exit rules. Therefore, PreM holds for rule (ii). For recursive rule (iii), the validity of PreM can be illustrated by adding an additional goal to the rule which will express the pre-application of the extrema constraint γ onto $T_\gamma(I)$, where γ is first applied to I. The recursive rule (iii) of question (c) can be re-written by adding an additional drop-in goal **\is_max((BasicSub),Time)/,** which will produce the following rule:

timeForBasic(AssPart, Time) <- assembly(AssPart, BasicSub,_), timeForBasic(BasicSub, Time),
\is_max((BasicPart), Time)/, is_max((AssPart),Time).

The bold goal in the above rule is the drop-in goal.

Thus, adding the drop-in goal corresponds to pre-applying the γ constraint (in our case: the max constraint) to the argument I in ($T_\gamma(I)$). For PreM to hold, we must have $T_\gamma(\gamma(I)) = T_\gamma(I)$ which tells that our drop-in goal has not changed the ICO mapping specified by the original recursive endo-max rule. In our example, the γ (max) constraint is applied to the pair (AssPart, Time) that produces the head of the recursive rule timeForBasic(AssPart, Time). This means our Time is maximized for each value of AssPart. The drop-in goal inserted after the goal timeForBasic(BasicSub, Time) is \is_max((BasicSub),Time)/. PreM is proven once we prove that the addition of this goal does not change the ICO mapping defined by the rule.

The validity of such a PreM can be inferred by the idea of Functional Dependencies that hold on the equivalent relational views of the predicates in the rule body. We will use the constraint based functional dependencies on tuples for the same. In our example, let a relation R(AssPart, BasicSub,Time) be a natural join of timeForBasic(SubPart,Time) and assembly(AssPart,SubPart,_). By using this, we can say that if a tuple t which belongs to R satisfies the max-constraint is_max((AssPart),Time), then R contains no tuple with the same AssPart-value and a larger Time-value.

This is depicted by $\text{AssPart} \xrightarrow{\max} \text{Time}$. This signifies that in our example, there is no other pair of AssPart and Time in the final result with the same AssPart and a larger Time. The following MVDs hold for all tuples in R: BasicPart -- >> Time and BasicPart -- >> AssPart. Therefore, by mixed transitivity, for any tuple R that satisfies the constraint $\text{AssPart} \xrightarrow{\max} \text{Time}$ also satisfies $\text{BasicPart} \xrightarrow{\max} \text{Time}$. Thus, if R' is the set of tuples in R that have the property $\text{AssPart} \xrightarrow{\max} \text{Time}$, R' also satisfies $\text{BasicPart} \xrightarrow{\max} \text{Time}$. Therefore, the drop-in constraint does not change R' or the mapping defined by it, since applying a constraint to a relation that already satisfied it does not change the relation.

Thus, we can conclude that Pre-mappability holds for question (c).

Since, PreM property holds for recursive rules of question (c), the extreme atoms, renamed *howsoon,* along with the atoms in the naïve fixpoint computation constitute the perfect model for the original program.

e) Write a Deal program to count the number of different basic parts used in each assembly:

```
% assembly(Part:string, Subpart:string, Qty:integer)
assembly('bike', 'frame', 1).
assembly('bike', 'wheel', 2).
assembly('frame', 'top_tube', 1).
assembly('frame', 'down_tube', 1).
assembly('frame', 'head_tube', 1).
assembly('frame', 'seat_mast', 1).
assembly('frame', 'seat_stay', 2).
assembly('frame', 'chain_stay', 2).
assembly('frame', 'fork', 1).
assembly('wheel', 'spoke', 36).
```

```
assembly('wheel', 'nipple', 36).
assembly('wheel', 'rim', 1).
assembly('wheel', 'hub', 1).
assembly('wheel', 'tire', 1).


% part_cost(Basic_part:string, Supplier:string, Cost:float, Time:integer)
part_cost('top_tube', 'cinelli', 20.0, 14).
part_cost('top_tube', 'columbus', 15.0, 6).
part_cost('down_tube', 'columbus', 10.0, 6).
part_cost('head_tube', 'cinelli', 20.0, 14).
part_cost('head_tube', 'columbus', 15.0, 6).
part_cost('seat_mast', 'cinelli', 20.0, 6).
part_cost('seat_mast', 'cinelli', 15.0, 14).
part_cost('seat_stay', 'cinelli', 15.0, 14).
part_cost('seat_stay', 'columbus', 10.0, 6).
part_cost('chain_stay', 'columbus', 10.0, 6).
part_cost('fork', 'cinelli', 40.0, 14).
part_cost('fork', 'columbus', 30.0, 6).
part_cost('spoke', 'campagnolo', 0.6, 15).
part_cost('nipple', 'mavic', 0.1, 3).
part_cost('hub', 'campagnolo', 31.0, 5).
part_cost('hub', 'suntour', 18.0, 14).
part_cost('rim', 'mavic', 50.0, 3).
part_cost('rim', 'araya', 70.0, 1).% assembly(Part:string, Subpart:string, Qty:integer)
assembly('bike', 'frame', 1).
assembly('bike', 'wheel', 2).
assembly('frame', 'top_tube', 1).
assembly('frame', 'down_tube', 1).
assembly('frame', 'head_tube', 1).
assembly('frame', 'seat_mast', 1).
assembly('frame', 'seat_stay', 2).
assembly('frame', 'chain_stay', 2).
assembly('frame', 'fork', 1).
assembly('wheel', 'spoke', 36).
assembly('wheel', 'nipple', 36).
assembly('wheel', 'rim', 1).
assembly('wheel', 'hub', 1).
assembly('wheel', 'tire', 1).


% part_cost(Basic_part:string, Supplier:string, Cost:float, Time:integer)
part_cost('top_tube', 'cinelli', 20.0, 14).
part_cost('top_tube', 'columbus', 15.0, 6).
part_cost('down_tube', 'columbus', 10.0, 6).
part_cost('head_tube', 'cinelli', 20.0, 14).
part_cost('head_tube', 'columbus', 15.0, 6).
part_cost('seat_mast', 'cinelli', 20.0, 6).
```

```
part_cost('seat_mast', 'cinelli', 15.0, 14).
part_cost('seat_stay', 'cinelli', 15.0, 14).
part_cost('seat_stay', 'columbus', 10.0, 6).
part_cost('chain_stay', 'columbus', 10.0, 6).
part_cost('fork', 'cinelli', 40.0, 14).
part_cost('fork', 'columbus', 30.0, 6).
part_cost('spoke', 'campagnolo', 0.6, 15).
part_cost('nipple', 'mavic', 0.1, 3).
part_cost('hub', 'campagnolo', 31.0, 5).
part_cost('hub', 'suntour', 18.0, 14).
part_cost('rim', 'mavic', 50.0, 3).
part_cost('rim', 'araya', 70.0, 1).

basic_subparts(BP, BP) <- part_cost(BP, _, _, _).
basic_subparts(P, BP) <- assembly(P, SP, _), basic_subparts(SP, BP).

export basic_subparts(X,Y).

different_Parts(P, countd<BP>) <- basic_subparts(P, BP), assembly(P, _, _).

export different_Parts(P,Y).
query different_Parts(P,Y).
```

```
root@LAPTOP-POR90PGJ:/mnt/c/Sneha/Studies/UCLA/Classes/Q3Spring2018/CS240A# ./runfile.sh DeALS-0.9.jar Qe.deal
Executing query 'different_Parts(P, Y).'.
Query Id = different_Parts(P,Y).1524187664409
Query Form = different_Parts(P, Y).
Return Status = SUCCESS
Execution time = 15ms
different_Parts(bike, 11).
different_Parts(frame, 7).
different_Parts(wheel, 4).
3 results
```

f) Write a Deal program to count the total number of basic parts used in each assembly.

```
% assembly(Part:string, Subpart:string, Qty:integer)
assembly('bike', 'frame', 1).
assembly('bike', 'wheel', 2).
assembly('frame', 'top_tube', 1).
assembly('frame', 'down_tube', 1).
assembly('frame', 'head_tube', 1).
assembly('frame', 'seat_mast', 1).
assembly('frame', 'seat_stay', 2).
assembly('frame', 'chain_stay', 2).
assembly('frame', 'fork', 1).
assembly('wheel', 'spoke', 36).
assembly('wheel', 'nipple', 36).
assembly('wheel', 'rim', 1).
assembly('wheel', 'hub', 1).
assembly('wheel', 'tire', 1).
```

```
% part_cost(Basic_part:string, Supplier:string, Cost:float, Time:integer)
part_cost('top_tube', 'cinelli', 20.0, 14).
part_cost('top_tube', 'columbus', 15.0, 6).
part_cost('down_tube', 'columbus', 10.0, 6).
part_cost('head_tube', 'cinelli', 20.0, 14).
part_cost('head_tube', 'columbus', 15.0, 6).
part_cost('seat_mast', 'cinelli', 20.0, 6).
part_cost('seat_mast', 'cinelli', 15.0, 14).
part_cost('seat_stay', 'cinelli', 15.0, 14).
part_cost('seat_stay', 'columbus', 10.0, 6).
part_cost('chain_stay', 'columbus', 10.0, 6).
part_cost('fork', 'cinelli', 40.0, 14).
part_cost('fork', 'columbus', 30.0, 6).
part_cost('spoke', 'campagnolo', 0.6, 15).
part_cost('nipple', 'mavic', 0.1, 3).
part_cost('hub', 'campagnolo', 31.0, 5).
part_cost('hub', 'suntour', 18.0, 14).
part_cost('rim', 'mavic', 50.0, 3).
part_cost('rim', 'araya', 70.0, 1).% assembly(Part:string, Subpart:string, Qty:integer)
assembly('bike', 'frame', 1).
assembly('bike', 'wheel', 2).
assembly('frame', 'top_tube', 1).
assembly('frame', 'down_tube', 1).
assembly('frame', 'head_tube', 1).
assembly('frame', 'seat_mast', 1).
assembly('frame', 'seat_stay', 2).
assembly('frame', 'chain_stay', 2).
assembly('frame', 'fork', 1).
assembly('wheel', 'spoke', 36).
assembly('wheel', 'nipple', 36).
assembly('wheel', 'rim', 1).
assembly('wheel', 'hub', 1).
assembly('wheel', 'tire', 1).
```

```
% part_cost(Basic_part:string, Supplier:string, Cost:float, Time:integer)
part_cost('top_tube', 'cinelli', 20.0, 14).
part_cost('top_tube', 'columbus', 15.0, 6).
part_cost('down_tube', 'columbus', 10.0, 6).
part_cost('head_tube', 'cinelli', 20.0, 14).
part_cost('head_tube', 'columbus', 15.0, 6).
part_cost('seat_mast', 'cinelli', 20.0, 6).
part_cost('seat_mast', 'cinelli', 15.0, 14).
part_cost('seat_stay', 'cinelli', 15.0, 14).
part_cost('seat_stay', 'columbus', 10.0, 6).
part_cost('chain_stay', 'columbus', 10.0, 6).
```

```
part_cost('fork', 'cinelli', 40.0, 14).
part_cost('fork', 'columbus', 30.0, 6).
part_cost('spoke', 'campagnolo', 0.6, 15).
part_cost('nipple', 'mavic', 0.1, 3).
part_cost('hub', 'campagnolo', 31.0, 5).
part_cost('hub', 'suntour', 18.0, 14).
part_cost('rim', 'mavic', 50.0, 3).
part_cost('rim', 'araya', 70.0, 1).

basic_subparts(BasicP, BasicP, 1) <- part_cost(BasicP, _, _, _).
basic_subparts(Part, BasicP, Q) <- assembly(Part, SubP, Q1), basic_subparts(SubP, BasicP,
Q2), Q=Q1*Q2.

export basic_subparts(X,Y,Q).

distinct_Parts(Part) <- assembly(Part,_,_).

export distinct_Parts(P).

totalParts(Part, sum<Q>) <- basic_subparts(Part, BasicP, Q), distinct_Parts(Part).

export totalParts(Part,Q).
query totalParts(Part,Q).
```

```
root@LAPTOP-POR90PGJ:/mnt/c/Sneha/Studies/UCLA/Classes/Q3Spring2018/CS240A# ./runfile.sh DeALS-0.9.jar Qf.deal
Executing query 'totalParts(Part, Q).'.
Query Id = totalParts(Part,Q).1524187470497
Query Form = totalParts(Part, Q).
Return Status = SUCCESS
Execution time = 11ms
totalParts(bike, 157).
totalParts(frame, 9).
totalParts(wheel, 74).
3 results
```