CompSci 516
Data Intensive Computing Systems

Lecture 13
Review of RC and SQL

Instructor: Sudeepa Roy
Why should we care about RC

• RC is declarative, like SQL, and unlike RA (which is operational)

• Gives foundation of database queries in first-order logic
  – you cannot express all aggregates in RC, e.g. cardinality of a relation or sum (possible in extended RA and SQL)
  – still can express conditions like “at least two tuples” (or any constant)

• RC expression may be much simpler than SQL queries
  – and easier to check for correctness than SQL
  – power to use $\forall$ and $\Rightarrow$
  – then you can systematically go to a “correct” SQL query
Useful Logical Equivalences

\[ \forall x \, P(x) = \neg \exists x \, \neg P(x) \]

\[ \neg (P \lor Q) = \neg P \land \neg Q \]
\[ \neg (P \land Q) = \neg P \lor \neg Q \]

\[ \text{Similarly, } \neg (\neg P \lor Q) = P \land \neg Q \text{ etc.} \]

\[ A \implies B = \neg A \lor B \]
Drinker Category 1

Find drinkers that frequent some bar that serves some beer they like.

FIRST TRY THE EXAMPLES YOURSELF BEFORE SEEING THE ANSWER IN THE FOLLOWING SLIDE

UNDERSTAND THE QUESTIONS

UNDERSTAND THE DIFFERENCE IN ANSWERS

Acknowledgement : examples and slides by Profs. Balazinska and Suciu and the [GUW] book
Drinker Category 1

Find drinkers that frequent some bar that serves some beer they like.

\[ Q(x) = \exists y. \exists z. \text{Frequents}(x, y) \land \text{Serves}(y, z) \land \text{Likes}(x, z) \]

a shortcut for
\{x | \exists Y \in \text{Frequents} \ Z \in \text{Serves} \ W \in \text{Likes} (T.\text{drinker} = x \land T.\text{bar} = Z.\text{bar} \land W.\text{beer} = \ldots.) \}

The difference is that in the first one, one variable = one attribute
in the second one, one variable = one tuple \((\text{Tuple RC})\)
Both are equivalent and feel free to use the one that is convenient to you
Find drinkers that frequent \textit{some} bar that serves \textit{some} beer they like.

\[ Q(x) = \exists y. \exists z. \text{Frequents}(x, y) \land \text{Serves}(y, z) \land \text{Likes}(x, z) \]

Find drinkers that frequent \textit{only} bars that serves \textit{some} beer they like.

\[ Q(x) = \ldots \]
Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

Drinker Category 2

Find drinkers that frequent some bar that serves some beer they like.

\[ Q(x) = \exists y. \exists z. \text{Frequents}(x, y) \land \text{Serves}(y, z) \land \text{Likes}(x, z) \]

Find drinkers that frequent only bars that serves some beer they like.

\[ Q(x) = \forall y. \text{Frequents}(x, y) \Rightarrow (\exists z. \text{Serves}(y, z) \land \text{Likes}(x, z)) \]
Drinker Category 3

Find drinkers that frequent some bar that serves some beer they like.

\[ Q(x) = \exists y. \exists z. \text{Frequents}(x, y) \land \text{Serves}(y, z) \land \text{Likes}(x, z) \]

Find drinkers that frequent only bars that serves some beer they like.

\[ Q(x) = \forall y. \text{Frequents}(x, y) \Rightarrow (\exists z. \text{Serves}(y, z) \land \text{Likes}(x, z)) \]

Find drinkers that frequent some bar that serves only beers they like.

\[ Q(x) = \ldots \]
Drinker Category 3

Find drinkers that frequent some bar that serves some beer they like.

\[ Q(x) = \exists y. \exists z. \text{Frequents}(x, y) \land \text{Serves}(y, z) \land \text{Likes}(x, z) \]

Find drinkers that frequent only bars that serves some beer they like.

\[ Q(x) = \forall y. \text{Frequents}(x, y) \Rightarrow (\exists z. \text{Serves}(y, z) \land \text{Likes}(x, z)) \]

Find drinkers that frequent some bar that serves only beers they like.

\[ Q(x) = \exists y. \text{Frequents}(x, y) \land \forall z. (\text{Serves}(y, z) \Rightarrow \text{Likes}(x, z)) \]
Likes(drinker, beer)
Frequents(drinker, bar)
Serves(bar, beer)

**Drinker Category 4**

Find drinkers that frequent some bar that serves some beer they like.

\[ Q(x) = \exists y. \exists z. \text{Frequents}(x, y) \land \text{Serves}(y, z) \land \text{Likes}(x, z) \]

Find drinkers that frequent only bars that serves some beer they like.

\[ Q(x) = \forall y. \text{Frequents}(x, y) \Rightarrow (\exists z. \text{Serves}(y, z) \land \text{Likes}(x, z)) \]

Find drinkers that frequent some bar that serves only beers they like.

\[ Q(x) = \exists y. \text{Frequents}(x, y) \land \forall z. (\text{Serves}(y, z) \Rightarrow \text{Likes}(x, z)) \]

Find drinkers that frequent only bars that serves only beer they like.

\[ Q(x) = \ldots \]
Drinker Category 4

Find drinkers that frequent some bar that serves some beer they like.

\[ Q(x) = \exists y. \exists z. \text{Frequents}(x, y) \land \text{Serves}(y, z) \land \text{Likes}(x, z) \]

Find drinkers that frequent only bars that serves some beer they like.

\[ Q(x) = \forall y. \text{Frequents}(x, y) \Rightarrow (\exists z. \text{Serves}(y, z) \land \text{Likes}(x, z)) \]

Find drinkers that frequent some bar that serves only beers they like.

\[ Q(x) = \exists y. \text{Frequents}(x, y) \land \forall z. (\text{Serves}(y, z) \Rightarrow \text{Likes}(x, z)) \]

Find drinkers that frequent only bars that serves only beer they like.

\[ Q(x) = \forall y. \text{Frequents}(x, y) \Rightarrow \forall z. (\text{Serves}(y, z) \Rightarrow \text{Likes}(x, z)) \]
Query: Find drinkers that like some beer (so much) that they frequent all bars that serve it

\[ Q(x) = \exists y. \text{Likes}(x, y) \land \forall z. (\text{Serves}(z, y) \Rightarrow \text{Frequents}(x, z)) \]
From RC to SQL

Query: Find drinkers that like some beer so much that they frequent all bars that serve it

\[ Q(x) = \exists y. \text{Likes}(x, y) \land \forall z. (\text{Serves}(z, y) \Rightarrow \text{Frequents}(x, z)) \]

\[ \equiv Q(x) = \exists y. \text{Likes}(x, y) \land \forall z. (\neg \text{Serves}(z, y) \lor \text{Frequents}(x, z)) \]

Step 1: Replace \( \forall \) with \( \exists \) using de Morgan’s Laws

\[ Q(x) = \exists y. \text{Likes}(x, y) \land \neg \exists z. (\text{Serves}(z, y) \land \neg \text{Frequents}(x, z)) \]
From RC to SQL

\[ Q(x) = \exists y. \text{Likes}(x, y) \land \neg \exists z. (\text{Serves}(z, y) \land \neg \text{Frequents}(x, z)) \]

Step 2: Translate into SQL

```
SELECT DISTINCT L.drinker
FROM Likes L
WHERE not exists
  (SELECT S.bar
   FROM Serves S
   WHERE L.beer=S.beer
   AND not exists (SELECT *
                    FROM Frequents F
                    WHERE F.drinker=L.drinker
                    AND F.bar=S.bar))
```
The “correct” intermediate steps

• Write the query in RC
• If you have a variable under “negation”, also add the “domain”, i.e. where the variable appears without a negation
  – e.g. if you have \( \neg \text{H}(x, y) \) for a subquery,
  – where \( x \) and \( y \) can only come from a relation \( R(x, y) \)
  – make it \( R(x, y) \land \neg \text{H}(x, y) \)

  – This is to make the query “safe” and “domain independent” – we will discuss this when we do Datalog
  – Intuitively, if you are trying to find “sailors that do not satisfy some criteria” you have to specify the domain of sailors, say from the sailor table, otherwise you are looking at an infinite space
The “correct” intermediate step

Make all subqueries with negation domain independent
i.e. say where x is coming from

\[ Q(x) = \exists y. \text{Likes}(x, y) \land \neg \exists z. (\text{Likes}(x, y) \land \text{Serves}(z, y) \land \neg \text{Frequents}(x, z)) \]

which can be simplified to the previous query

```sql
SELECT DISTINCT L.drinker FROM Likes L
WHERE not exists
    (SELECT * FROM Likes L2, Serves S
     WHERE L2.drinker=L.drinker and L2.beer=L.beer
     and L2.beer=S.beer
     and not exists (SELECT * FROM Frequents F
                     WHERE F.drinker=L2.drinker
                     and F.bar=S.bar))
```