

Deductive Databases
Summer Term 2018
 Prof. Dr. W. May

1. Unit: Kalkül I

Self-contained subformulas (i.e., formulas in RANF) will also be needed for translating complex queries into Datalog programs. The transformation into RANF works in the same intuitive way as done in the DB lecture before knowing the relational calculus at all. For complex cases (and for the computer), a systematic formal procedure helps.

Exercise 1 (Kalkül: Sichere, Wertebereichsunabhängige Anfragen) Check for the following queries whether they are in SRNF (give $rr(G)$ for each of their subformulas).

For the formulas that are in RANF:

- check whether the formulas are in RANF. If not, give an equivalent formula in RANF.
- Give equivalent expressions in the relational algebra and in SQL (develop the SQL expressions both from the original formula and from the RANF formula).

- a) $F(X, Y, Z) = p(X, Y) \wedge (q(Y) \vee r(Z))$,
 b) $F(X, Y) = p(X, Y) \wedge (q(Y) \vee r(X))$,
 c) $F(X, Y) = p(X, Y) \wedge \neg \exists Z : r(Y, Z)$,
 d) $F(X) = p(X) \wedge \exists Y : (q(Y) \wedge \neg r(X, Y))$,
 e) $F(X) = p(X) \wedge \neg \exists Y : (q(Y) \wedge \neg r(X, Y))$
 f) $F(X, Y) = \exists V : (r(V, X) \wedge \neg s(X, Y, V)) \wedge \exists W : (r(W, Y) \wedge \neg s(Y, X, W))$

a) $p(X, Y) \wedge (q(Y) \vee r(Z))$:

G	$rr(G)$
$p(x, y)$	X, Y
$q(Y)$	Y
$r(Z)$	Z
$q(Y) \vee r(Z)$	$\{Y\} \cap \{Z\} = \emptyset$
$p(X, Y) \wedge (q(Y) \vee r(Z))$	$\{X, Y\} \cup \emptyset = \{X, Y\}$

Since $free(F) = \{X, Y, Z\} \neq \{X, Y\} = rr(F)$, F is not in SRNF (and thus also not in RANF).
 F is not domain-independent: for \mathcal{S} with $\mathcal{S}(p) = \{1, a\}$ and $\mathcal{S}(q) = \{(a)\}$ and $\mathcal{S}(r) = \emptyset$ and domain \mathcal{D} is the answer set $\{X \mapsto 1, Y \mapsto a, Z \mapsto d \mid d \in \mathcal{D}\}$.

b) $p(X, Y) \wedge (q(Y) \vee r(X))$:

G	$rr(G)$
$p(x, y)$	X, Y
$q(Y)$	Y
$r(X)$	X
$q(Y) \vee r(X)$	$\{Y\} \cap \{X\} = \emptyset$
$p(X, Y) \wedge (q(Y) \vee r(X))$	$\{X, Y\} \cup \emptyset = \{X, Y\}$

Since $free(F) = \{X, Y\} = rr(F)$, F is in SRNF.

F is not in RANF since the disjunction $q(Y) \vee r(X)$ is not self-contained.

F can easily be expressed in SQL (with $P(P_1, P_2), Q(Q_1), R(R_1)$):

```
SELECT P1, P2
FROM P
WHERE P2 in (SELECT Q1 FROM Q)
OR P1 in (SELECT R1 FROM R)
```

The equivalent expression in the relational algebra is

$$(P \bowtie_{P_2=Q_1} Q) \cup (P \bowtie_{P_1=R_1} R).$$

This is also obtained when translating from SRNF to RANF with “push-into-or”:

$$(p(X, Y) \wedge q(Y)) \vee (p(X, Y) \wedge r(Z))$$

and then translates as usual to the relational algebra.

c) $F(X, Y) = p(X, Y) \wedge \neg \exists Z : r(Y, Z)$:

G	$rr(G)$
$p(X, Y)$	X, Y
$r(Y, Z)$	Y, Z
$\exists Z : r(Y, Z)$	Y
$\neg \exists Z : r(Y, Z)$	\emptyset
$p(X, Y) \wedge \neg \exists Z : r(Y, Z)$	X, Y

Since $free(F) = \{X\} = rr(F)$, F is in SRNF.

All subformulas are self-contained.

F can easily be expressed in SQL (with $P(P_1, P_2), R(R_1, R_2)$):

```
SELECT P1, P2
FROM P
WHERE P2 NOT IN (SELECT R2 FROM R)
```

The equivalent expression in the relational algebra is

$$P \bowtie ((\pi[P_2](P) - \pi[R_1](R))).$$

The standard translation that uses the enumeration formula for the active domain (here: those that occur in P and R) reads as:

$$P \bowtie ((\pi[P_1](P) \cup \pi[P_2](P) \cup \pi[R_1](R) \cup \pi[R_2](R)) - \pi[R_1](R)).$$

d) $F(X) = p(X) \wedge \exists Y : (q(Y) \wedge \neg r(X, Y))$:

G	$rr(G)$
$p(X)$	X
$q(Y)$	Y
$r(X, Y)$	X, Y
$\neg r(X, Y)$	\emptyset
$q(Y) \wedge \neg r(X, Y)$	Y
$\exists Y : q(Y) \wedge \neg r(X, Y)$	\emptyset
$p(X) \wedge \exists Y : q(Y) \wedge \neg r(X, Y)$	X

Since $free(F) = \{X\} = rr(F)$, F is in SRNF.

F is not in RANF since the subformula $G = \exists Y : q(Y) \wedge \neg r(X, Y)$ is not self-contained: for the body $H = q(Y) \wedge \neg r(X, Y)$ there is $free(H) = \{X, Y\} \supsetneq \{Y\} = rr(H)$ (note that the SAFE criterion from the lecture would already detect H as the problem).

F can easily be expressed in SQL (with $P(P_1), Q(Q_1), R(R_1, R_2)$):

```
SELECT P1
FROM P
WHERE EXISTS (SELECT Q1
```

```

FROM Q
WHERE (P1,Q1) NOT IN (SELECT R1,R2 FROM R)

```

The equivalent expression in the relational algebra is

$$\pi[P_1]((P \times Q) - \rho[R_1 \rightarrow P_1, R_2 \rightarrow P_2]R).$$

This is also obtained when translating from SRNF to RANF with “push-into-exist”:

$$F(X) = \exists Y : (p(X) \wedge q(Y) \wedge \neg r(X, Y)),$$

and then translates as usual to the relational algebra.

This corresponds to the (simpler) SQL query

```

SELECT P1
FROM P, Q
WHERE (P1,Q1) NOT IN (SELECT R1,R2 FROM R)

```

e) This formula is the pattern of the relational division, $r \div q$. Es ist äquivalent zu $F(X) = p(X) \wedge \forall Y : (q(Y) \rightarrow r(X, Y))$.

$$F(X) = p(X) \wedge \neg \exists Y : (q(Y) \wedge \neg r(X, Y)),$$

G	$rr(G)$
$p(X)$	X
$q(Y)$	Y
$r(X, Y)$	X, Y
$\neg r(X, Y)$	\emptyset
$q(Y) \wedge \neg r(X, Y)$	Y
$\exists Y : q(Y) \wedge \neg r(X, Y)$	\emptyset
$\neg \exists Y : q(Y) \wedge \neg r(X, Y)$	\emptyset
$p(X) \wedge \neg \exists Y : q(Y) \wedge \neg r(X, Y)$	X

Since $free(F) = \{X\} = rr(F)$, F is in SRNF.

F is \neg -as in (d)– not in RANF since the subformula $G = \exists Y : q(Y) \wedge \neg r(X, Y)$ is not self-contained.

F can easily be expressed in SQL (with $P(P_1), Q(Q_1), R(R_1, R_2)$):

```

SELECT P1
FROM P
WHERE NOT EXISTS (SELECT Q1
                  FROM Q
                  WHERE (P1,Q1) NOT IN (SELECT R1,R2 FROM R))

```

The equivalent expression in the relational algebra is

$$P - \pi[P_1]((P \times Q) - \rho[R_1 \rightarrow P_1, R_2 \rightarrow P_2](R)).$$

This is also obtained when translating from SRNF to RANF with “push-into-not-exist”:

$$F(X) = p(X) \wedge \neg \exists Y : (p(X) \wedge q(Y) \wedge \neg r(X, Y)),$$

and then translates as usual to the relational algebra.

f) This is an example for a conjunction, where none of the conjuncts is self-contained:

$$F(X, Y) = \exists V : (r(V, X) \wedge \neg s(X, Y, V)) \wedge \exists W : (r(W, Y) \wedge \neg s(Y, X, W))$$

G	$rr(G)$
$r(V, X)$	X, V
$s(X, Y, V)$	X, Y, V
$\neg s(X, Y, V)$	\emptyset
$r(V, X) \wedge \neg s(X, Y, V)$	X, V
$\exists V : (r(V, X) \wedge \neg s(X, Y, V))$	X
$r(W, Y)$	W, Y
$s(Y, X, W)$	X, Y, W
$\neg s(Y, X, W)$	\emptyset
$r(W, Y) \wedge \neg s(Y, X, W)$	W, Y
$\exists W : (r(W, Y) \wedge \neg s(Y, X, W))$	Y
$(\dots) \wedge (\dots)$	X, Y

Since $free(F) = \{X, Y\} = rr(F)$, F is in SRNF.

F is not in RANF since the subformulas $\exists V : (r(V, X) \wedge \neg s(X, Y, V))$ and $\exists W : (r(W, Y) \wedge \neg s(Y, X, W))$ are not self-contained (again, the problem is located inside each of the subformulas, as SAFE would complain about).

F can easily be expressed in SQL (with $P(P_1), Q(Q_1), R(R_1, R_2)$):

```
SELECT rv.R2, rw.R2
FROM R rv, R rw
WHERE NOT EXISTS (SELECT * FROM S
                  WHERE S1=rv.R2 and S2=rw.R2 and S3=rv.R1)
AND NOT EXISTS (SELECT * FROM S
                WHERE S1=rw.R2 and S2=rv.R2 and S3=rw.R1)
```

or

```
SELECT rv.R2, rw.R2
FROM R rv, R rw
WHERE NOT (rv.R2, rw.R2, rv.R1 IN (SELECT * FROM S))
AND NOT (rw.R2, rv.R2, rw.R1 IN (SELECT * FROM S))
```

The equivalent expression in the relational algebra is ... not that easy.

Thus, F has to be transformed from SRNF to RANF by moving the first conjunct into the second by “push-into-exists” (or the vice versa, the final result is the same):

$$\exists W : \exists V : (r(V, X) \wedge \neg s(X, Y, V)) \wedge (r(W, Y) \wedge \neg s(Y, X, W))$$

Flatten existential quantifiers, flatten conjunction:

$$\exists V, W : B(X, Y, V, W)$$

with $B = (r(V, X) \wedge r(W, Y) \wedge \neg s(X, Y, V) \wedge \neg s(Y, X, W))$

is self-contained with $free(B) = \{V, W, X, Y\} = rr(B)$

According to the transformation algorithm given in the lecture, the following has to be done:

- build the (XYV) component of B , subtract s ,
- in parallel build the (XYW) component of B , subtract s ,
- these are the triples of bindings that “survive”,
- join them,

- and project to:

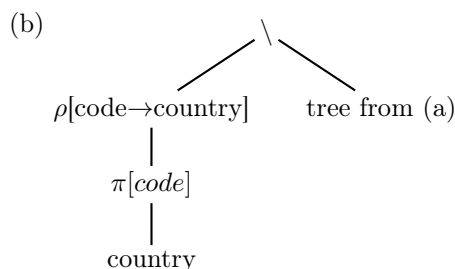
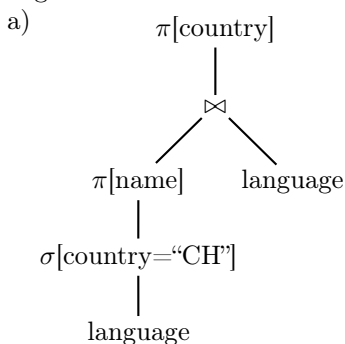
$$\begin{aligned} \pi[X, Y](& (\pi[X, Y, V](\rho[R_1 \rightarrow V, R_2 \rightarrow X](r) \times \rho[R_1 \rightarrow W, R_2 \rightarrow Y](r)) \\ & - \rho[S_1 \rightarrow X, S_2 \rightarrow Y, R_2 \rightarrow V](s)) \\ \bowtie (& (\pi[X, Y, W](\rho[R_1 \rightarrow V, R_2 \rightarrow X](r) \times \rho[R_1 \rightarrow W, R_2 \rightarrow Y](r)) \\ & - \rho[S_1 \rightarrow Y, S_2 \rightarrow X, R_2 \rightarrow W](s))) \end{aligned}$$

Exercise 2 (Relationale Anfragen an Mondial: Schweizer Sprachen) Give expressions in the relational calculus for the following queries against the Mondial database. Compare with the same queries in the relational Algebra and in SQL.

- All codes of countries, in which some languages is spoken that is also spoken in Switzerland.
- All codes of countries, in which only languages are spoken that are not spoken in Switzerland.
- All codes of countries, in which only languages are spoken that are also spoken in Switzerland.
- All codes of countries in which all languages that are spoken in Switzerland are also spoken.

- $F(C) = \exists L, Perc1, Perc2 : (\text{language}('CH', L, Perc1) \wedge \text{language}(C, L, Perc2))$
- $F(C) = \exists CN, A, Pop, Cap, CapP : (\text{country}(CN, C, A, Pop, Cap, CapP) \wedge \neg \exists L, Perc1, Perc2 : (\text{language}('CH', L, Perc1) \wedge \text{language}(C, L, Perc2)))$

Algebra:

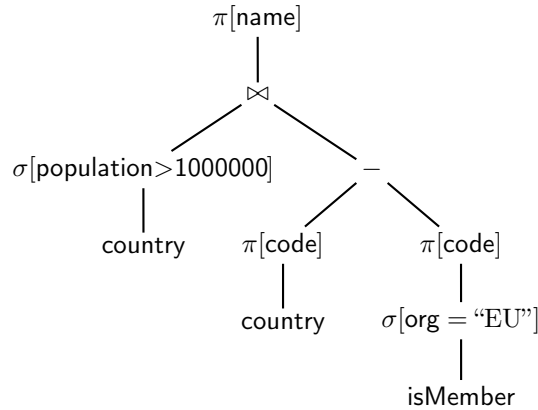


- $F(C) = (\exists CN, A, Pop, Cap, CapP : \text{country}(CN, C, A, Pop, Cap, CapP)) \wedge \neg \exists L, Perc1 : (\text{language}(C, L, Perc1) \wedge \neg \exists Perc1 : \text{language}('CH', L, Perc2))$
- $F(C) = (\exists CN, A, Pop, Cap, CapP : \text{country}(CN, C, A, Pop, Cap, CapP)) \wedge \forall L : ((\exists Perc1 : \text{language}('CH', L, Perc1)) \rightarrow (\exists Perc2 : \text{language}(C, L, Perc2)))$

Exercise 3 (RANF to Algebra – Minus) Give expressions in the relational algebra and in the relational calculus for the query “Full names of all countries that have more than 1000000 inhabitants and are not member of the EU”.

Check whether the calculus expression is in SRNF and RANF, and transform it into the relational algebra. Compare the result with the algebra expression.

A straightforward algebra expression is



The calculus expression is

$$F(N) = \exists C, Cap, CapProv, A, Pop : \\ (\text{country}(N, C, Cap, CapProv, A, Pop) \wedge Pop > 1000000 \wedge \neg \exists T : \text{isMember}(C, \text{"EU"}, T)) .$$

It is in SRNF, it is safe range, and it is in RANF. Recall that for the subformula $\neg \exists T : \text{isMember}(C, \text{"EU"}, T)$, RANF requires $rr(\exists T : \text{isMember}(C, \text{"EU"}, T)) = free(\exists T : \text{isMember}(C, \text{"EU"}, T)) = \{C\}$ which is the case.

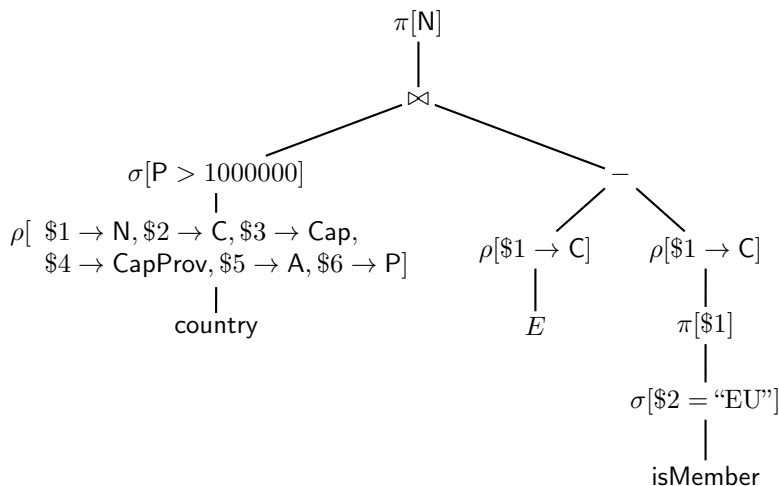
For the relational algebra,

$$\begin{aligned} \text{isMember}(C, \text{"EU"}, T) &\Rightarrow \rho[\$1 \rightarrow C, \$3 \rightarrow T](\pi[\$1, \$3](\sigma[\$2 = \text{"EU"}](\text{isMember}))) \\ \exists T : \text{isMember}(C, \text{"EU"}, T) &\Rightarrow \pi[\$1](\rho[\$1 \rightarrow C, \$3 \rightarrow T](\pi[\$1, \$3](\sigma[\$2 = \text{"EU"}](\text{isMember})))) \\ &= \rho[\$1 \rightarrow C](\pi[\$1](\sigma[\$2 = \text{"EU"}](\text{isMember}))) \end{aligned}$$

For $\neg \exists T : \text{isMember}(C, \text{"EU"}, T)$, let the expression E denote the algebra expression that enumerates all values of the active domain. With this,

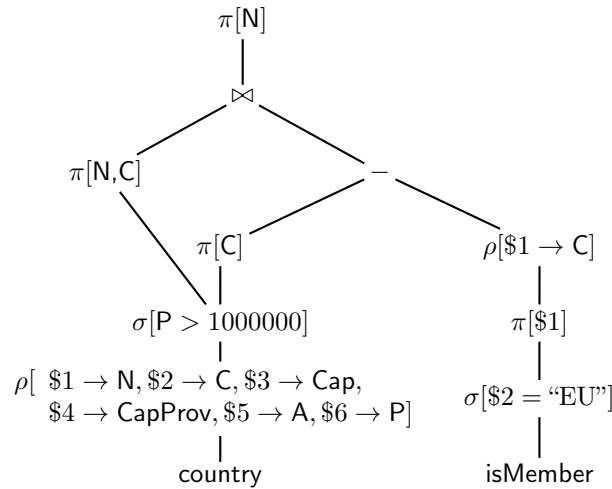
$$\neg \exists T : \text{isMember}(C, \text{"EU"}, T) \Rightarrow \rho[\$1 \rightarrow C](E) - \rho[\$1 \rightarrow C](\pi[\$1](\sigma[\$2 = \text{"EU"}](\text{isMember})))$$

Altogether, the whole query translates to

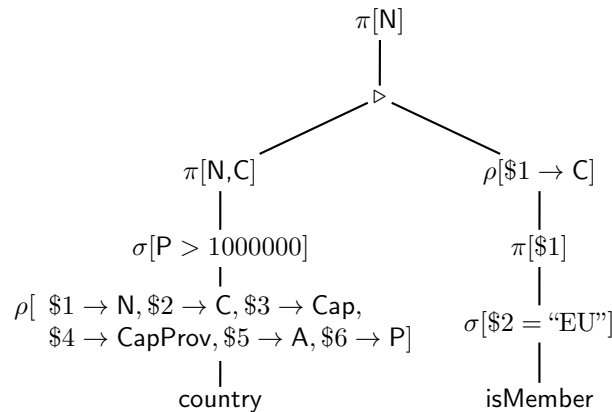


Obviously, the term $\rho[\$1 \rightarrow C](E)$ can be replaced by $\rho[\$2 \rightarrow C](\pi[\$2](\text{country}))$ which enumerates a superset of all values of C that can result from the left subtree.

Instead, also $\rho[\$2 \rightarrow C](\pi[\$2](\sigma[\$6 > 1000000](\text{country})))$ is sufficient, which makes the left subtree (nearly) unnecessary. From it, only the full name must still be obtained.



Another possibility is the anti-join \triangleright (which is one of the built-in operators of internal algebras):



Exercise 4 (Division: Äquivalenz von Algebra und Kalkül) For the relational algebra, the division operator has been introduced as a derived operator (cf. lecture “Databases”). Consider the relation schemata $r(A, B)$ and $s(B)$.

$$r \div s = \{\mu \in \text{Tuple}(A) \mid \{\mu\} \times s \subseteq r\} = \pi[A](r) \setminus \pi[A](\pi[A](r) \times s \setminus r).$$

Derive a query in the relational calculus from the left-hand side, and prove the equivalence with the right-hand side.

The left-hand side expression: the set of all possible tuples over a A is described by $F(X) = \text{ADOM}(X)$. The remaining task is then easy: for all values Y in S , the combination of X and Y must be in R :

$$F(X) = \text{ADOM}(X) \wedge \forall Y : (s(Y) \rightarrow r(X, Y)) .$$

Here, it is obvious that instead $ADOM(X)$, the consideration can be restricted to the A -values of R :

$$F(X) = \exists Z : r(X, Z) \wedge \forall Y : (s(Y) \rightarrow r(X, Y)) .$$

The query is not in SRNF. It is equivalent to

$$F(X) = \exists Z : r(X, Z) \wedge \neg \exists Y : (s(Y) \wedge \neg r(X, Y)) ,$$

which is in SRNF (thus, domain-independent), but not in RANF.

Transformation to RANF (“push-into-not-exists”):

$$F(X) = \exists Z : r(X, Z) \wedge \neg \exists Y : (\exists Z_2 : r(X, Z_2)) \wedge s(Y) \wedge \neg r(X, Y)$$

Derivation of the algebra expression:

F	$Algebra$
$(\exists Z : r(X, Z)) \wedge s(Y) \wedge \neg r(X, Y)$	$(\pi[A](r) \times s) \setminus r$
$\exists Y : (\exists Z : r(X, Z)) \wedge s(Y) \wedge \neg r(X, Y)$	$\pi[A]((\pi[A](r) \times s) \setminus r)$ (the expression has the format A)
$\exists Z : r(X, Z)$	$\pi[A](r)$ (has again the format A)
$F(X)$ as above	$\pi[A](r) \setminus \pi[A]((\pi[A](r) \times s) \setminus r)$

... is exactly the right-hand side.

Exercise 5 (Kalkül: Gruppierung und Aggregation) Define a syntactical extension for the relational calculus, that allows to use aggregate functions similar to the `GROUP BY` functionality of SQL.

For this, consider only aggregate functions as simple applications over single attributes like $\max(\text{population})$, but not more complex expressions like $\max(\text{population/area})$.

- What is the result of an aggregate function, and how can it be used in the calculus?
- Which inputs does an aggregate function have?
- how can this input be obtained from the database?

Give a calculus expression for the query “For each country give the name and the total number of people living in its cities”.

The result is a number. It can be bound to a variable or it can be used in a comparison. Thus, the aggregate function is to be considered as a term (whose evaluation yields a value).

The immediate input to an aggregate function is a set/list of values, over which the aggregate is computed (sum, count, ...).

This list can be obtained as results of a (sub)formula (similar to a correlated subquery) with a free variable.

The results are grouped by zero, one or more free variables of the subquery. Usually, these also occur in other literals outside the aggregation.

$$X = \text{agg-op}\{\text{var } [group-by-vars]; \text{subq-fml}\}$$

where in *subq-fml* the *group-by-vars* and *var* have free occurrences. E.g.,

$$F(CN, SumCityPop) = \\ \exists C, A, P, Cap, CapProv : country(CN, C, A, P, Cap, CapProv) \wedge \\ SumCityPop = \text{sum}\{CityPop [C]; \\ \exists CtyN, CtyProv, L1, L2 : city(CtyN, CtyProv, C, CityPop, L1, L2)\}$$

groups by C , computes the sum over $CityPop$ and binds the value to $SumCityPop$.

Comments:

- a similar syntax is used in F-Logic;
- the usage in XSB is similar, but the user has to program it more explicitly:
 - the list is created by the Prolog predicate “bagof”;
 - the aggregation operation over the list must be programmed in the common Prolog style for handling a list.

Exercise 6 (Kalkül→Algebra) Consider the relation schemata $R(A, B)$, $S(B, C)$ und $T(A, B, C)$.

a) Give an equivalent algebra expression for the following safe relational calculus expression:

$$F_1(X, Y) = T(Y, a, Y) \wedge (R(a, X) \vee S(X, c)) \wedge \neg T(a, X, Y)$$

Proceed as shown in the lecture for the equivalence proof.

b) Simplify the expression.

c) Extend the expression from (a) to

$$F_2(Y) = \exists X : (F_1(X, Y) \wedge X > 3)$$

a) First, consider each of the three conjuncts (denoted as F_2 , F_1 and F_3) separately:

The literal $F_1(Y) = T(Y, a, Y)$ corresponds to the subexpression

$$E_1 = \rho[A \rightarrow Y](\pi[A](\sigma[(A = C) \wedge (B = a)](T))) .$$

The subformula $F_2(X) = R(a, X) \vee S(X, c)$ corresponds to the expression

$$E_2 = \rho[B \rightarrow X](\pi[B](\sigma[A = a](R))) \cup \rho[B \rightarrow X](\pi[B](\sigma[C = c](S))) .$$

The negated literal $F_3(X, Y) = \neg T(a, X, Y)$ corresponds to the expression

$$E_4 = \rho[B \rightarrow X, C \rightarrow Y](\pi[B, C](\sigma[A = a](T)))$$

The expression corresponding to $F_3(X, Y)$ is then

$$E_3 = \rho[\$1 \rightarrow X, \$2 \rightarrow Y](ADOM^2) - \rho[B \rightarrow X, C \rightarrow Y](\pi[B, C](\sigma[A = a](T)))$$

where $ADOM^2 = ((\pi[A](R) \cup \pi[B](R) \cup \pi[B](S) \cup \pi[C](S) \cup \pi[A](T) \cup \pi[B](T) \cup \pi[C](T)) \times (\pi[A](R) \cup \pi[B](R) \cup \pi[B](S) \cup \pi[C](S) \cup \pi[A](T) \cup \pi[B](T) \cup \pi[C](T)))$ contains all 2-tuples of values from the database.

Thus, $E = E_1 \bowtie E_2 \bowtie (ADOM^2 - E_4)$ is the complete algebra expression.

b) Simplify: E_1 and E_2 have no variable/column in common, thus it can be simplified as $(E_1 \times E_2) \bowtie (ADOM^2 - E_4)$. Both subterms bind X and Y , thus, $ADOM^2$ can be omitted, obtaining $E' = (E_1 \times E_2) - E_4$.

c) The additional comparison is expressed as a selection, and the $\exists X$ quantification is expressed as a projection to Y :

$$\pi[Y](\sigma[X > 3](E'))$$