1. Take two relations
$r(A_1, \ldots, A_m, B_1, \ldots, B_n)$
$s(B_1, \ldots, B_n, C_1, \ldots, C_p)$
We will rename $B_1, \ldots, B_n$ in s to new attributes $D_1, \ldots, D_n$. Then we take a cross product with r, use select to enforce $B_i = D_i$, and project away $D_1, \ldots, D_n$.

$$\Pi_{A_1,\ldots,A_m,B_1,\ldots,B_n,C_1,\ldots,C_p}\left(\sigma_{B_1=D_1 \wedge \cdots \wedge B_n=D_n}\left(r \times \rho_{B_1\to D_1,\ldots,B_n\to D_n}(s)\right)\right)$$

Example: boats(BNAME, COLOR, BID) ⋈ reserves(BID, SID, DAY)

$$\Pi_{BNAME,COLOR,BID,SID,DAY}\left(\sigma_{BID=D1}\left(boats \times \rho_{BID\to D1}(reserves)\right)\right)$$

2. The key observation is that any expression with just SPJU is monotone: If we have an SPJU expression E on $r_1, \ldots, r_n$, and we add a tuple to any of $r_1, \ldots, r_n$, the answer is either the same or grows in size:

$$E(r_1, \ldots, r_i + \{t\}, \ldots, r_n) \supseteq E(r_1, \ldots, r_i, \ldots, r_n).$$

This condition can easily be violated by ~~once~~ queries involving difference or division.

Need to show monotonicity of SPJU

S: $E = \sigma_F(E_1)$. Then $\sigma_F(E_1 + \{t\}) = \sigma_F(E_1) + \sigma_F(\{t\}) \supseteq E$, since $\sigma_F(\{t\})$ contains 0 or 1 tuples

P: $E = \Pi_X(E_1)$. Then $\Pi_X(E_1 + \{t\})$ is either:
$\Pi_X(E_1)$ if $t[X] \in \Pi_X(E_1)$
$\Pi_X(E_1) + \{t[X]\}$ ~~otherwise~~ otherwise
In either case, it contains E

J: $E = E_1 \bowtie E_2$ Consider $(E_1 + \{t\}) \bowtie E_2$ (The E_2 case is similar)
$(E_1 + \{t\}) \bowtie E_2 = (E_1 \bowtie E_2) + (\{t\} \bowtie E_2) \supseteq$
$E_1 \bowtie E_2 = E$, since $\{t\} \bowtie E_2$ contains 0 or 1 tuples

2. (Cont)

U: $E = E_1 \cup E_2$ Consider $(E_1 + \{t\}) \cup E_2$ (E_2 case similar)

$(E_1 + \{t\}) \cup E_2 = (E_1 \cup E_2) + \{t\} \supseteq =$

~~$E \cup$~~ $E + \{t\} \supseteq E$

Since each of SPJU is monotone, any expression built up from them is monotone, and hence can't express, for example, $r - s$.

3. i. safe ii. safe iii. unsafe: E only appears negatively
iv. safe v. unsafe: C only appears in the head

4 i. algebra: $\pi_{Pcourse}(Prereq \bowtie \rho_{Pcourse \to Course, Course \to Other}(\sigma_{Course='MTH427'}(Prereq)))$

Datalog: res(PC) :- Prereq(PC, C), Prereq(C, 'MTH 427').

Domain Calc: $\{p: Pcourse \mid \exists c: Course\ (Prereq(p,c) \wedge Prereq(c, 'MTH427')\}$

Tuple Calc: $\{P \mid \exists R \in Prereq, \exists S \in Prereq\ (P.Pcourse = R.Pcourse \wedge R.Course = S.Pcourse \wedge S.Course = 'MTH427')\}$

ii algebra: $MajorReq \div \pi_{Course}(\sigma_{Degree='BA'}(UnivReq))$

domain Calc: $\{m: Major \mid \forall c: Course\ (UnivReq(c, 'BA') \to MajorReq(c, m))\}$

tuple Calc $\{T \mid \forall R \in UnivReq\ (R.Degree = 'BA' \to \exists S \in MajorReq\ (R.Course = S.Course \wedge R.Major = T.Major)\}$

Datalog Missing(m) :- MajorReq(x, m), UnivReq(c, 'BA'), ¬MajorReq(c, m).

Res(m) :- MajorReq(x, m), ¬Missing(m).

5i.

$\Pi_{Course}(\sigma_{Major='CS'}(MajorReq \bowtie PreReq)) -$

~~$\Pi_{Course}(\sigma_{Degree='BA'}(UnivReq))$~~

$\rho_{Course \to PCourse}(\Pi_{Course}(\sigma_{Degree='BA'}(UnivReq)) \cup \Pi_{Course}(\sigma_{Major='CS'}(MajorReq)))$

ii)

```
CReq(c) :- MajorReq(c, 'EE').
CReq(c) :- UnivReq(c, 'BA').
Res(p, c) :- CReq(p), PreReq(p,c), c='CS302'.
```

6i. I realize that I should have added one requirement to SPJ queries, namely that selects can only be simple, since something like $\sigma_{A=1 \vee A=2}(r)$ is essentially a union in disguise and requires 2 Datalog rules.

The main issue with solutions to this question came in the SPJ → S1RDWON direction. (Thanks to V. Megler for the acronym). People showed how to translate each of S, P + J into S1RDWON, but weren't explicit about how to reduce the set of rules to a single rule. This has to be done with care, to avoid sharing variables.

Consider

```
Res(x) :- r(x,y), s(y,z).
r(x,y) :- t(x,y), q(y,z).
```

If we simply substitute the right side of r for r in the Res rule we get

Res(x) :- t(x,y), q(y,z), s(y,z).

any the repeated z imposes a constraint not in the original program. We need to start with an r rule with distinct variables from the Res rule:

r(x1, y1) :- t(x1, y1), q(y1, z1).

We can then substitute and equate:

Res(x) :- t(x1, y1), q(y1, z1), s(y, z), x=x1, y=y1.

If we want, we can simplify to

Res(x) :- t(x, y), q(y, z1), s(y, z).

no sharing problem.

We describe the translation of a SPJ query Q to a SIRDWON program P by induction on the number of operators in P. We will assume at each point that the result variables in P are named the same as the result attributes in Q.

Basis: 0 operators in Q. So Q = r. If r has schema A1, ..., Am, then P is

Res(A1, ..., Am) :- r(A1, ..., Am). This rule is safe

Induction: Assume there is a translation for q operators, and consider the q+1st operator.

Case S1: $Q = \sigma_{A_i=c}(Q1)$. Let Res1(A1, ..., Am) :- L1, L2, ..., Lp. Be the SIRDWON programs for Q1. Then the program P for Q is Res(A1, ..., Am) :- L1, L2, ..., Lp, Ai=c.

Case S2: $Q = \sigma_{A_i=A_j}(Q1)$. Then P is

Res(A1, ..., Am) :- L1, L2, ..., Lp, Ai=Aj.

Safe if Res1 is safe.

6: (cont)

Case P: $Q = \pi_{A_1 \ldots A_i}(Q1)$. Let Res1 for Q1 be as before.

Then the program for Q is

$$Res(A1, \ldots, Ai) :- L_1, L_2, \ldots, L_p.$$

This rule is save if Res1 is safe, because the body didn't change and there are fewer variables in the head.

Case J: $Q = Q1 \bowtie Q2$ Let the programs for Q1 & Q2 be

$$Res1(A_1, \ldots, A_m, B_1, \ldots, B_n) :- L_1, L_2, \ldots, L_p.$$

$$Res2(B_1, \ldots, B_n, C_1, \ldots, C_\ell) :- M_1, M_2, \ldots, M_k.$$

Rename any variables ~~not in~~ in the body of Res2 that are not in the head to be distinct from any variables in Res1.

The program P for Q is

$$Res(A_1, \ldots, A_m, B_1, \ldots, B_n, C_1, \ldots, C_\ell) :- L_1, L_2, \ldots L_p, M_1, M_2, \ldots, M_k.$$

This rule is safe because every head variable of Res appears in some L_i or M_j. The body is safe if Res1 and Res2 are safe.

Example: Consider relations r(A B) and s(B C), with query

$$Q = \sigma_{B=5}(r) \bowtie \pi_{AC}(r \bowtie s)$$

Q_1, Q_2, Q_3, Q_4, Q_5, Q_6

Resi for Qi

Res1(A, B) :- r(A, B).
Res2(A, B) :- r(A, B).
Res3(B, C) :- s(B, C)
Res4(A, B, C) :- r(A, B), s(B, C).
Res5(A, B) :- r(A, B), B=5

~~rename B in Res4.~~

Res6(A, C) :- r(A, B), s(B, C).

rename B in Res6:

Res6(A, C) :- r(A, B1), s(B1, C).

Res(A, B, C) :- r(A, B), B=5, r(A, B1), s(B1, C).

6i (cont)

We can go from SIRDWON → SPJ as follows.

Suppose we have program P:

Res $(x_1, \ldots, x_m)$:- $L_1, L_2, \ldots, L_p$.

Where each L_i is a positive literal or atomic formula $\binom{x=y}{x=c}$

Assume the L_i's are ordered so the literals come before the atomic formulae. * Note that L_1 must be a literal, for safety. We will have an expression E_i that corresponds to $L_1, L_2, \ldots, L_i$.

Basis L_1 must be $r(y_1, \ldots, y_n)$ for relation $r(A_1, \ldots, A_n)$.

E_1 is $\rho_{A1 \to y_1, \ldots, A_n \to y_n}(r)$

Induction. Have E_i for $L_1, \ldots, L_i$. Consider L_{i+1}

Case 1: L_i is $s(w_1, \ldots, w_k)$ for relation $s(B_1, \ldots, B_k)$.

$E_{i+1} = \rho_{B1 \to w_1, \ldots, B \to w_k}(s) \bowtie E_i$

Case 2: L_i is $x=y$

$E_{i+1} = \sigma_{x=y}(E_i)$

Case 3: L_i is $x=c$

$E_{i+1} = \sigma_{x=c}(E_i)$

The expression for the whole rule is

$E = \pi_{x_1 \ldots x_m}(E_p)$

* I also need to assume that no literal contains a constant c. If it does, replace c by new variable z and add z=c. Repeated variable x in a literal is replaced by $z_1 + z_2$ plus $z_1 = z_2$.

6i (cont)

Example on $r(A\ B\ C)$, $s(B, C, D)$

$Res(x, y) :- r(x, z, w), s(z, 6, u), r(u, y, y).$

~~Get rid~~ Handle the 6 and the repeated y

$Res(x, y) :- r(x, z, w), s(z, v, u), r(u, p, q), v=6, y=p, y=q.$

L_1 L_2 L_3 L_4 L_5 L_6

E $\pi_{xy}($

E_6 $\sigma_{y=q}($

E_5 $\sigma_{y=p}($

E_4 $\sigma_{v=6}($

E_3 $\rho_{A\to u, B\to p, C\to q}(r) \bowtie$

E_2 $\rho_{B\to z, C\to v, D\to u}(s) \bowtie$

E_1 $\rho_{A\to x, B\to z, C\to w}(r)$

)

)

)

)

6ii One thing we can't handle in 1-rule Datalog is difference with a join. Consider $r(A), s(A), u(A)$

Then ~~Res~~ $r - (s \bowtie u)$ can be written in 2 rules:

$Res(x) :- r(x), \neg Res1(x).$

$Res1(y) :- s(y), u(y).$

But not in ~~one~~. one.

Note: $Res(x) :- r(x), \neg(s(x), u(x)).$ is not legal Datalog and

$Res(x) :- r(x), \neg s(x), \neg u(x).$ is not equivalent.