Updated 3/30/2022

Topic 1: Data models and query languages
Unit 4: Datalog (continued)
Lecture 11

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp22)

https://northeastern-datalab.github.io/cs7240/sp22/

2/22/2022

Pre-class conversations

- Last class recapitulation: Datalog w/ negation: PTIME vs NPC
- Time for feedback
- Will look at scribes this week

- today:
  - Practice with Datalog, and evaluation strategies (incl. view maintenance)
  - Starting with T2: complexity of query evaluation

What do empty bodies or heads mean in ASP?

```
A :- B, not C.
```

Think of the head as a disjunction, body as conjunction

$0 \vee A \Leftarrow 1 \wedge B \wedge \neg C$

DLV = "DataLog with Disjunction (=V)"

Empty body:

```
A.
```

?

Empty head:

```
:- B, not C.
```

?

What do empty bodies or heads mean in ASP?

```
A :- B, not C.
```

Think of the head as a disjunction, body as conjunction

$0 \vee A \Leftarrow 1 \wedge B \wedge \neg C$

DLV = "DataLog with Disjunction (=V)"

Empty body:

```
A.
```

$A \Leftarrow 1$

Empty body describes a fact: "A" needs to be true. Also in Datalog

Empty head:

```
:- B, not C.
```

$0 \Leftarrow B \wedge \neg C$

Empty heads describes a constraint: "B and not C" must not be true in any model. Emtpy head describes a condition in the body which leads to contradiction (false)

Complexity and Expressive Power of Logic Programming

EVGENY DANTSIN
Roosevelt University, Chicago, IL, USA
THOMAS EITER, GEORG GOTTLOB
Vienna University of Technology, Austria
AND
ANDREI VORONKOV
University of Manchester, United Kingdom

Note that every stratified P has a unique stable model, and its stratified and stable semantics coincide. Unstratified rules increase complexity.

Informally, *disjunctive logic programming* (*DLP*) extends logic programming by adding disjunction in the rule heads, in order to allow more natural and flexible knowledge representation. For example,

$$male(X) \vee female(X) \leftarrow person(X)$$

naturally represents that any person is either male or female.

THEOREM 5.7. ([*Marek and Truszczyński 1991; Bidoit and Froidevaux 1991*]). *Given a propositional normal logic program P, deciding whether $SM(P) \neq \emptyset$ is* NP*-complete.*

THEOREM 5.8. (*Marek and Truszczyński 1991*; *Schlipf 1995b*; *Kolaitis and Papadimitriou 1991*]). *Propositional logic programming with negation under SMS is* co-NP*-complete. Datalog with negation under SMS is data complete for* co-NP *and program complete for* co-NEXPTIME.

Example for NP-complete problem: Boolean satisfiability problem: "given a Boolean formula, is it satisfiable" (i.e. is there an input for which the formula outputs true)?

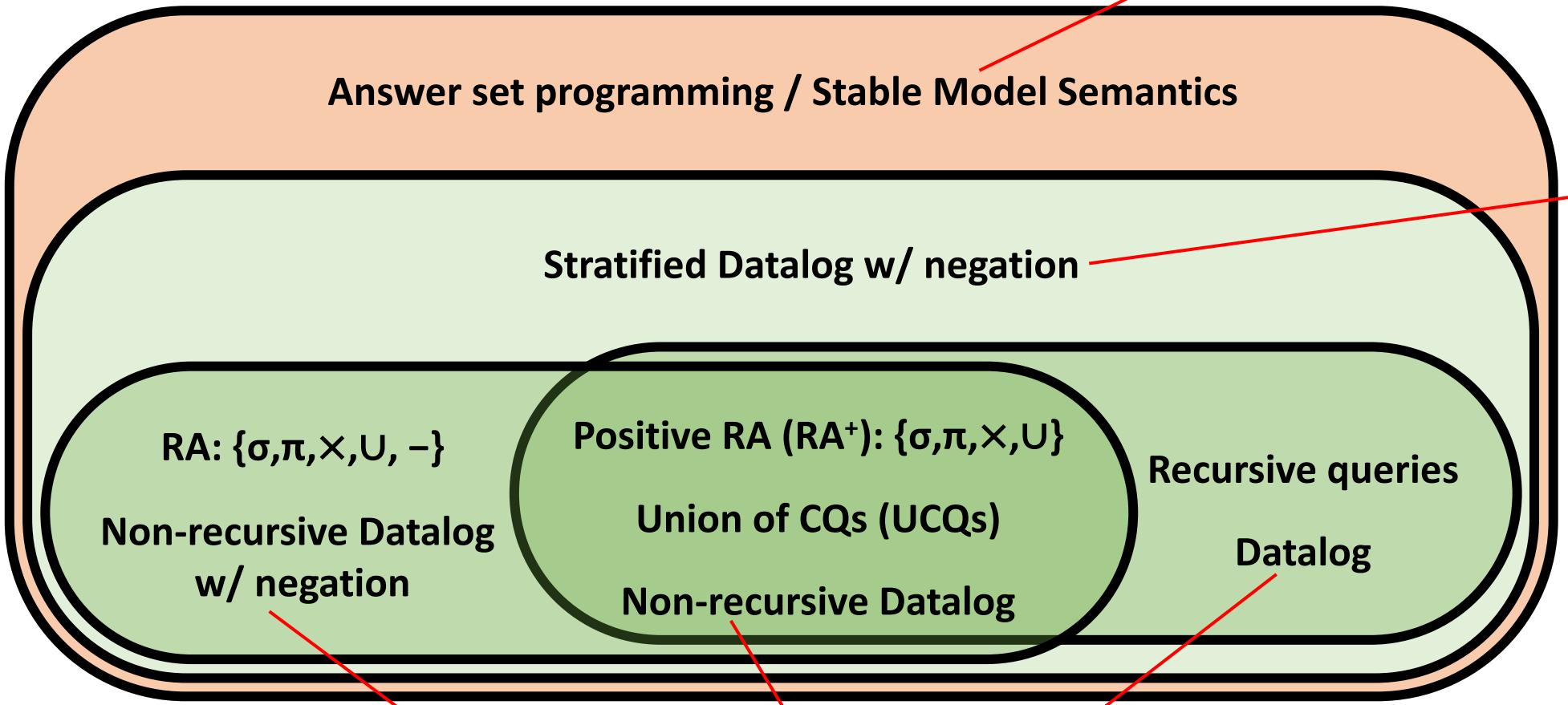
Example for co-NP problem: the complementary problem asks: "given a Boolean formula, is it unsatisfiable" (i.e. do all possible inputs to the formula output false)?

Modeling problems beyond the class NP with ASP is possible to some extent. Namely, when disjunctions are allowed in the heads of rules, every decision problem in the class Σ_2^P can be modeled in a uniform way by a finite program (Dantsin et al. 2001). However, modeling problems beyond NP with ASP is complicated and the generate-define-test approach is no longer sufficient in general. Additional techniques such as *saturation* (Eiter and Gottlob 1995) are needed but they are difficult to use, and may introduce constraints that have no direct relation to constraints of the problem being modeled. As stated explicitly in (Gebser et al. 2011) "unlike the ease of common ASP modeling, [...] these techniques are rather involved and hardly usable by ASP laymen."

Dantsin, Eiter, Gottlob, Voronkov. "Complexity and expressive power of logic programming", ACM computing survesy, 2001. https://doi.org/10.1145/502807.502810
Amendola, Ricca, Truszczynski. "Beyond NP: Quantifying over Answer Sets", TPLP, 2019. https://doi.org/10.1017/S1471068419000140
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Hierarchy of expressiveness

ASP can express NP-complete problems
(For Turing-completeness, we would only have to add functions, i.e. the ability to create new values not previously found in the EDB)



can express all polynomial time queries on ordered databases relying on only information encoded in tables (e.g. excludes arithmetical functions)

Notice that Datalog and UCQs often assume an unordered domain and no built-in predicates. For equality, we assume here an ordered domain and built-in predicates (>,<,≤,≥,!=).

Outline: T1-4: Datalog

- Datalog
  - Datalog rules
  - Recursion
  - Semantics
  - Datalog$^{\neg}$: Negation, stratification
  - Datalog$^{\pm}$
  - Stable model semantics (Answer set programming)
  - Datalog vs. RA
  - Naive and Semi-naive evaluation (incl. Incremental View Maintenance)

RA to Datalog by examples: Union

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

R(A,B,C) ∪ S(D,E,F)

Datalog:

?

RA to Datalog by examples: Union

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

R(A,B,C) ∪ S(D,E,F)

Datalog:

Q(x,y,z) :- R(x,y,z)
Q(x,y,z) :- S(x,y,z)

IDB EDB

?

RA to Datalog by examples: Union

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

R(A,B,C) ∪ S(D,E,F)

Datalog:

Q(x,y,z) :- R(x,y,z)
Q(x,y,z) :- S(x,y,z)

IDB EDB

RA to Datalog by examples: Intersection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$R(A,B,C) \cap S(D,E,F)$

Datalog:

?

RA to Datalog by examples: Intersection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$R(A,B,C) \cap S(D,E,F)$

Datalog:

```
Q(x,y,z) :- R(x,y,z), S(x,y,z)
```

RA to Datalog by examples: Selection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$\sigma_{B='Alice' \wedge C>10}(R)$

Datalog:

?

RA to Datalog by examples: Selection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$\sigma_{B='Alice' \wedge C>10}(R)$

Datalog:

Q(x,y,z) :- R(x,y,z), y='Alice', z > 10

(also: Q(x,y,z) :- R(x,'Alice',z), z > 10)

RA to Datalog by examples: Selection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$\sigma_{B='Alice' \wedge C>10}(R)$

Datalog:

Q(x,y,z) :- R(x,y,z), y='Alice', z > 10

RA:

$\sigma_{B='Alice' \vee C>10}(R)$

?

RA to Datalog by examples: Selection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$\sigma_{B='Alice' \wedge C>10}(R)$

Datalog:

Q(x,y,z) :- R(x,y,z), y='Alice', z > 10

RA:

$\sigma_{B='Alice' \vee C>10}(R)$

Datalog:

Q(x,y,z) :- R(x,y,z), y='Alice'
Q(x,y,z) :- R(x,y,z), z > 10

RA to Datalog by examples: Projection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$\pi_{A}(R)$ $\pi_{\text{-B,C}}(R)$

Datalog:

?

RA to Datalog by examples: Projection

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$\pi_{A}(R)$ $\pi_{-B,C}(R)$

Datalog:

Q(x) :- R(x,y,z)

Q(x) :- R(x,_,_)

Underscore denotes an "anonymous variable".
Each occurrence of an underscore represents a different variable

RA to Datalog by examples: Equi-join

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$$\pi_{\text{-D,E}}(\ R \bowtie_{A=D \wedge B=E}\ S)$$

Datalog:

?

RA to Datalog by examples: Equi-join

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

$\pi_{-D,E}(R \bowtie_{A=D \wedge B=E} S)$

Datalog:

Q(x,y,z,w) :- R(x,y,z), S(x,y,w)

RA to Datalog by examples: Difference

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

R−S

Datalog:

?

RA to Datalog by examples: Difference

R(A,B,C)
S(D,E,F)
T(G,H)

RA:

R−S

Datalog¬: (we need to add negation)

Q(x,y,z) :- R(x,y,z), not S(x,y,z)

Outline: T1-4: Datalog

- Datalog
  - Datalog rules
  - Recursion
  - Semantics
  - Datalog$^{\neg}$: Negation, stratification
  - Datalog$^{\pm}$
  - Stable model semantics (Answer set programming)
  - Datalog vs. RA
  - Naive and Semi-naive evaluation (incl. Incremental View Maintenance)

Datalog Evaluation Algorithms

- Goal: preserve the efficiency of query optimizers, yet extend them to recursion
- Two general strategies we will discuss:
  - 1. Naive Datalog evaluation
  - 2. Semi-naive Datalog evaluation
- More powerful optimizations:
  - 3. Magic sets (which we will not cover, or may revisit later under "Topic 3: efficient query evaluation & factorized representations")

1. Naive Datalog evaluation

$P^{(t)}(x,y)$:- $A(x,y)$.
$P^{(t)}(x,y)$:- $A(x,z)$, $P^{(t-1)}(z,y)$.

$P^{(0)} := \emptyset$, $t:=0$
Repeat {
  $inc(t)$
  $P^{(t)}(x,y) := A(x,y) \cup \Pi_{xy}(A(x,z) \bowtie P^{(t-1)}(z,y))$
until $P^{(t)} = P^{(t-1)}$}

immediate consequence operator "T_P": $P^{(t)} = T_P(P^{(t-1)})$

- Problem: The same facts are discovered over and over again
- Goal: The semi-naive algorithm tries to reduce the number of facts discovered multiple times

Example

$P^{(t)}(x,y)$:- $A(x,y)$.
$P^{(t)}(x,y)$:- $A(x,z)$, $P^{(t-1)}(z,y)$.

A

1	2
2	3
3	4
4	5

$P^{(1)}$?

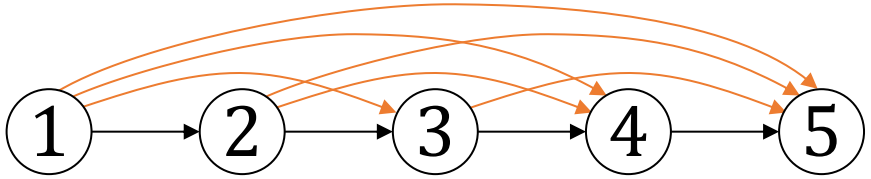
$P^{(2)}$?

$P^{(3)}$?

$P^{(4)}$?

Example

$P^{(t)}(x,y)$:- $A(x,y)$.
$P^{(t)}(x,y)$:- $A(x,z)$, $P^{(t-1)}(z,y)$.



A

1	2
2	3
3	4
4	5

$P^{(1)}$

1	2
2	3
3	4
4	5

1st

$P^{(2)}$

1	2
2	3
3	4
4	5
1	3
2	4
3	5

1st
2nd

$P^{(3)}$

1	2
2	3
3	4
4	5
1	3
2	4
3	5
1	4
2	5

1st
2nd

$P^{(4)}$

1	2
2	3
3	4
4	5
1	3
2	4
3	5
1	4
2	5
1	5

1st
2nd

Incremental View Maintentance

Background: Incremental View Maintenace

Let Q be a "view" computed by a single Datalog rule without recursion, thus a simple conjunctive query

$Q :- R_1, R_2, \ldots$

```
SELECT ...
FROM R1
NATURAL JOIN R2
NATURAL JOIN R3 ...
```

Add tuples to some of the relations:

$$R_1 \longleftarrow R_1 \cup \Delta R_1,\ R_2 \longleftarrow R_2 \cup \Delta R_2,\ \ldots$$

Then the view Q will also increase in size:

$$Q \longleftarrow Q \cup \Delta Q$$

Incremental view maintenance problem:
Compute ΔQ without having to recompute Q from scratch

Background: Incremental View Maintenace

Example 1:

Q(x,y) :- R(x,z), S(z,y)

ΔQ(x,y) :- ?

If R ⟵ R∪ΔR,
then what is ΔQ ?

Q

1	1
1	2

ΔQ ?

R

1	0
2	0

ΔR

S

0	1
0	2
1	1

Background: Incremental View Maintenace

Example 1:

Q(x,y) :- R(x,z), S(z,y)

ΔQ(x,y) :- ?

If R ⟵ R∪ΔR,
then what is ΔQ ?

Q

1	1
1	2

ΔQ

1	1
1	2

R

1	0

ΔR

2	0

S

0	1
0	2
1	1

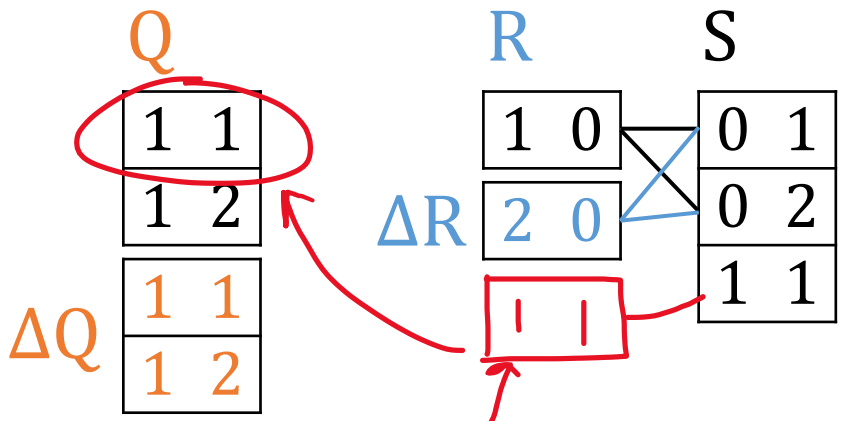
Background: Incremental View Maintenace

Example 1:

$Q(x,y) :- R(x,z), S(z,y)$

$\Delta Q(x,y) :- \Delta R(x,z), S(z,y)$

If $R \longleftarrow R \cup \Delta R$,
then what is ΔQ ?



(to be more precise: we still need to subtract Q:
$\Delta Q = \Delta R \bowtie S - Q$, e.g. for $\Delta R = (1,1)$. More on that later)

Relational Algebra:

$Q = R \times S$

$Q \cup \Delta Q = (R \cup \Delta R) \times S$

?

Background: Incremental View Maintenace

Example 1:

$Q(x,y) :- R(x,z), S(z,y)$

$\Delta Q(x,y) :- \Delta R(x,z), S(z,y)$

If $R \longleftarrow R \cup \Delta R$,
then what is ΔQ ?

Q	
1	1
1	2

ΔQ	
1	1
1	2

R	
1	0
2	0

ΔR

S	
0	1
0	2

(to be more precise: we still need to subtract Q:
$\Delta Q = \Delta R \bowtie S - Q$, e.g. for $\Delta R = (1,1)$. More on that later)

$z = x \cdot y$

$z+\Delta z = (x+\Delta x) \cdot y$

?

Relational Algebra:

$Q = R \times S$

$Q \cup \Delta Q = (R \cup \Delta R) \times S$

?

Background: Incremental View Maintenace

Example 1:

$Q(x,y) :- R(x,z), S(z,y)$

$\Delta Q(x,y) :- \Delta R(x,z), S(z,y)$

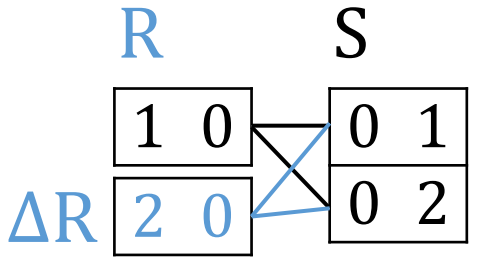
If $R \longleftarrow R \cup \Delta R$,
then what is ΔQ ?

Q

1	1
1	2

ΔQ

1	1
1	2



(to be more precise: we still need to subtract Q:
$\Delta Q = \Delta R \bowtie S - Q$, e.g. for $\Delta R = (1,1)$. More on that later)

Multiplication ⊗ distributes over Addition ⊕

$$z = x \cdot y$$
$$z+\Delta z = (x+\Delta x) \cdot y$$
$$z+\Delta z = (x \cdot y)+(\Delta x \cdot y)$$
$$z+\Delta z = z +(\Delta x \cdot y)$$
$$\Delta z = \Delta x \cdot y$$

Relational Algebra:

$$Q = R \times S$$
$$Q \cup \Delta Q = (R \cup \Delta R) \times S$$
$$Q \cup \Delta Q = (R \times S) \cup (\Delta R \times S)$$
$$Q \cup \Delta Q = Q \cup (\Delta R \times S)$$
$$\Delta Q = \Delta R \times S$$

Cartesian Product distributes over union ☺

Background: Incremental View Maintenace

Example 2:

Q(x,y) :- R(x,z), S(z,y)

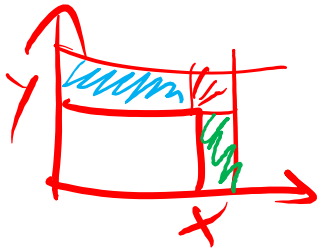
If R ⟵ R∪ΔR, and S ⟵ S∪ΔS,
then what is ΔQ ?

(as before, we ignore the subtraction of Q here)

?

$z = x \cdot y$

$z+\Delta z = (x+\Delta x)\cdot(y+\Delta y)$



?

Background: Incremental View Maintenace

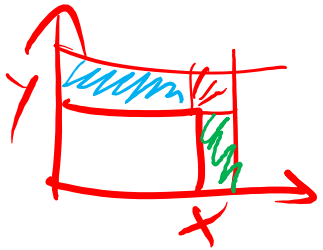
Example 2:

$Q(x,y) :- R(x,z), S(z,y)$

If $R \longleftarrow R \cup \Delta R$, and $S \longleftarrow S \cup \Delta S$,
then what is ΔQ ?

(as before, we ignore the subtraction of Q here)

?



$$z = x \cdot y$$
$$z+\Delta z = (x+\Delta x)\cdot(y+\Delta y)$$
$$z+\Delta z = (x\cdot y)+(\Delta x\cdot y)+(x\cdot\Delta y)+(\Delta x\cdot\Delta y)$$
$$z+\Delta z = z +(\Delta x\cdot y)+(x\cdot\Delta y)+(\Delta x\cdot\Delta y)$$
$$\Delta z = (\Delta x\cdot y)+(x\cdot\Delta y)+(\Delta x\cdot\Delta y)$$

Relational Algebra:

$$Q = R \times S$$
$$Q \cup \Delta Q = (R \cup \Delta R) \times (S \cup \Delta S)$$
$$Q \cup \Delta Q = (R \times S) \cup (\Delta R \times S) \cup (R \times \Delta S) \cup (\Delta R \times \Delta S)$$
$$Q \cup \Delta Q = Q \cup (\Delta R \times S) \cup (R \times \Delta S) \cup (\Delta R \times \Delta S)$$
$$\Delta Q = (\Delta R \times S) \cup (R \times \Delta S) \cup (\Delta R \times \Delta S)$$

Background: Incremental View Maintenace

Example 2:

$Q(x,y) :- R(x,z), S(z,y)$

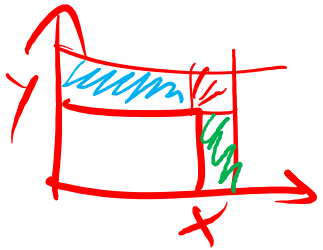
$\Delta Q(x,y) :- \Delta R(x,z), S(z,y)$

$\Delta Q(x,y) :- R(x,z), \Delta S(z,y)$

$\Delta Q(x,y) :- \Delta R(x,z), \Delta S(z,y)$

If $R \longleftarrow R \cup \Delta R$, and $S \longleftarrow S \cup \Delta S$,
then what is ΔQ ?

(as before, we ignore the subtraction of Q here)



$$z = x \cdot y$$
$$z+\Delta z = (x+\Delta x)\cdot(y+\Delta y)$$
$$z+\Delta z = (x\cdot y)+(\Delta x\cdot y)+(x\cdot\Delta y)+(\Delta x\cdot\Delta y)$$
$$z+\Delta z = z +(\Delta x\cdot y)+(x\cdot\Delta y)+(\Delta x\cdot\Delta y)$$
$$\Delta z = (\Delta x\cdot y)+(x\cdot\Delta y)+(\Delta x\cdot\Delta y)$$

Relational Algebra:

$$Q = R \times S$$
$$Q \cup \Delta Q = (R \cup \Delta R) \times (S \cup \Delta S)$$
$$Q \cup \Delta Q = (R \times S) \cup (\Delta R \times S) \cup (R \times \Delta S) \cup (\Delta R \times \Delta S)$$
$$Q \cup \Delta Q = Q \cup (\Delta R \times S) \cup (R \times \Delta S) \cup (\Delta R \times \Delta S)$$
$$\Delta Q = (\Delta R \times S) \cup (R \times \Delta S) \cup (\Delta R \times \Delta S)$$

Background: Incremental View Maintenace

Example 3:

$Q(x,y) :- R(x,z), R(z,y)$

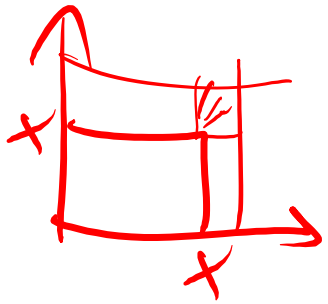
If $R \longleftarrow R \cup \Delta R$,
then what is ΔQ ?

(as before, we ignore the subtraction of Q here)

?

$z = x^2$

$z+\Delta z = (x+\Delta x)^2$



?

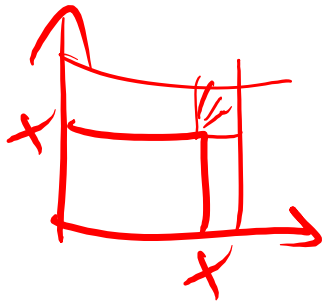
Background: Incremental View Maintenace

Example 3:

$Q(x,y) :- R(x,z), R(z,y)$

If $R \longleftarrow R \cup \Delta R$,
then what is ΔQ ?
(as before, we ignore the subtraction of Q here)

?



$$z = x^2$$
$$z+\Delta z = (x+\Delta x)^2$$
$$z+\Delta z = x^2+(\Delta x \cdot x)+(x \cdot \Delta x)+\Delta x^2$$
$$z+\Delta z = z +2x\Delta x+\Delta x^2$$
$$\Delta z = 2x\Delta x+\Delta x^2$$

Relational Algebra:

$$Q = R \times R$$
$$Q \cup \Delta Q = (R \cup \Delta R) \times (R \cup \Delta R)$$

?

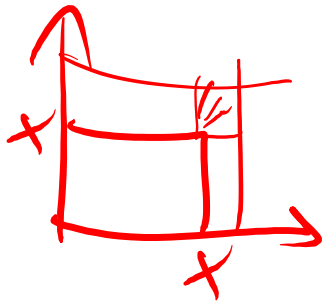
Background: Incremental View Maintenace

Example 3:

$Q(x,y) :- R(x,z), R(z,y)$

If $R \longleftarrow R \cup \Delta R$,
then what is ΔQ ?
(as before, we ignore the subtraction of Q here)

?



$$z = x^2$$
$$z+\Delta z = (x+\Delta x)^2$$
$$z+\Delta z = x^2+(\Delta x \cdot x)+(x \cdot \Delta x)+\Delta x^2$$
$$z+\Delta z = z +2x\Delta x+\Delta x^2$$
$$\Delta z = 2x\Delta x+\Delta x^2$$

Relational Algebra:

$$Q = R \times R$$
$$Q \cup \Delta Q = (R \cup \Delta R) \times (R \cup \Delta R)$$
$$Q \cup \Delta Q = (R \times R) \cup (\Delta R \times R) \cup (R \times \Delta R) \cup (\Delta R \times \Delta R)$$
$$Q \cup \Delta Q = Q \cup (\Delta R \times R) \cup (R \times \Delta R) \cup (\Delta R \times \Delta R)$$
$$\Delta Q = (\Delta R \times R) \cup (R \times \Delta R) \cup (\Delta R \times \Delta R)$$

Background: Incremental View Maintenace

Example 3:

$Q(x,y) :- R(x,z), R(z,y)$

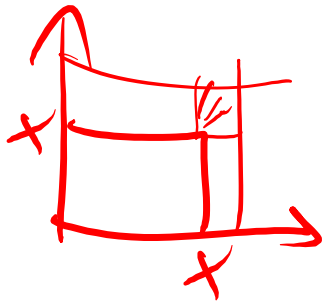
$\Delta Q(x,y) :- \Delta R(x,z), R(z,y)$

$\Delta Q(x,y) :- R(x,z), \Delta R(z,y)$

$\Delta Q(x,y) :- \Delta R(x,z), \Delta R(z,y)$

If $R \longleftarrow R \cup \Delta R$,
then what is ΔQ ?

(as before, we ignore the subtraction of Q here)



$$z = x^2$$
$$z+\Delta z = (x+\Delta x)^2$$
$$z+\Delta z = x^2+(\Delta x \cdot x)+(x \cdot \Delta x)+\Delta x^2$$
$$z+\Delta z = z +2x\Delta x+\Delta x^2$$
$$\Delta z = 2x\Delta x+\Delta x^2$$

Relational Algebra:

$$Q = R \times R$$
$$Q \cup \Delta Q = (R \cup \Delta R) \times (R \cup \Delta R)$$
$$Q \cup \Delta Q = (R \times R) \cup (\Delta R \times R) \cup (R \times \Delta R) \cup (\Delta R \times \Delta R)$$
$$Q \cup \Delta Q = Q \cup (\Delta R \times R) \cup (R \times \Delta R) \cup (\Delta R \times \Delta R)$$
$$\Delta Q = (\Delta R \times R) \cup (R \times \Delta R) \cup (\Delta R \times \Delta R)$$

Back to Datalog evaluation

2. Semi-Naive Datalog evaluation

Path(x,y) :- Arc(x,y).
Path(x,y) :- Arc(x,z), Path(z,y).

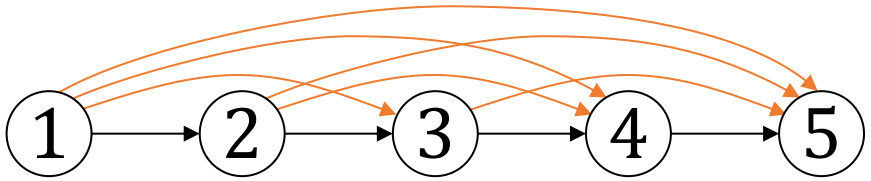
Recall the naive evaluation:

$$\text{Path}^{(0)} := \emptyset,\ t:=0$$
$$\text{Repeat}\ \{$$
$$\quad \text{inc}(t)$$
$$\quad \text{Path}^{(t)}(x,y) := \text{Arc}(x,y) \cup \Pi_{xy}(\text{Arc}(x,z) \bowtie \text{Path}^{(t-1)}(z,y))$$
$$\text{until}\ \text{Path}^{(t)} = \text{Path}^{(t-1)}\}$$

Semi-naive evaluation:

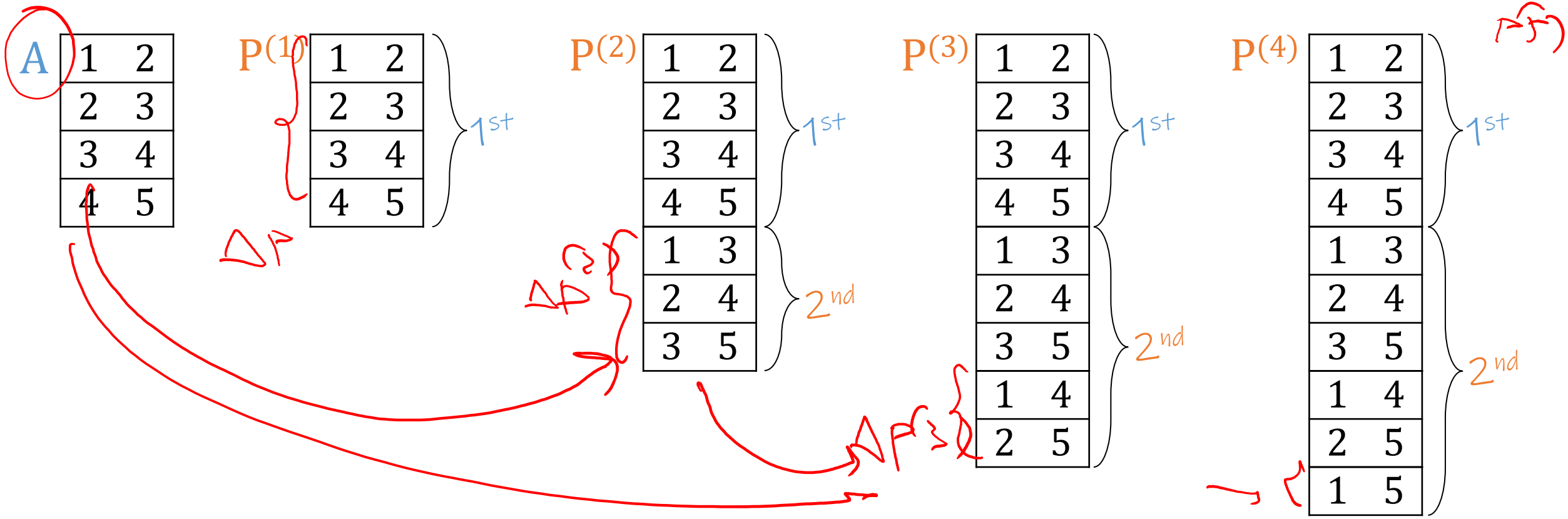
$$\text{Path} := \text{Arc}(x,z);\ \Delta\text{Path} := \text{Arc}(x,z)$$
$$\text{Repeat}\ \{$$
$$\quad \Delta\text{Path}(x,y) := \Pi_{xy}(\text{Arc}(x,z) \bowtie \Delta\text{Path}(z,y)) - \text{Path}(x,y)$$
$$\quad \text{Path} := \text{Path} \cup \Delta\text{Path}$$
$$\text{until}\ \Delta\text{Path} = \emptyset\}$$

Example



$P^{(t)}(x,y)$:- $A(x,y)$.
$P^{(t)}(x,y)$:- $A(x,z)$, $P^{(t-1)}(z,y)$.

$\Delta P^{(t)}(x,y)$:- $A(x,z)$, $\Delta P^{(t-1)}(z,y)$, not $P^{(t-1)}(x,y)$.
$P^{(t)}(x,y)$:- $P^{(t-1)}(x,y)$.
$P^{(t)}(x,y)$:- $\Delta P^{(t)}(x,y)$.



Datalog Summary

- EDB (extensional/base relations), IDB (intentional/derived relations)
- Datalog program = set of rules; base relations are also rules
- Datalog can be recursive
  - Stratified Datalog with negation still PTIME
  - Non-stratified Datalog: stable model semantics, ASP, can model NPC problems
- SQL has also been extended to express limited form of recursion
  - Using a recursive "with" clause, also called CTE (Common Table Expression)
  - Can only have a single IDB

For more details on recursion in SQL see https://www.postgresql.org/docs/14/queries-with.html#QUERIES-WITH-RECURSIVE

Outline: T1-4: Datalog

- Datalog
  - Datalog rules
  - Recursion
  - Semantics
  - Datalog$^{\neg}$: Negation, stratification
  - Datalog$^{\pm}$
  - Stable model semantics (Answer set programming)
  - Datalog vs. RA
  - Naive and Semi-naive evaluation (incl. Incremental View Maintenance)