

# DATABASE THEORY

## Lecture 14: Datalog Implementation

Markus Krötzsch

Knowledge-Based Systems

TU Dresden, 30th May 2022

More recent versions of this slide deck might be available.  
For the most current version of this course, see  
[https://iccl.inf.tu-dresden.de/web/Database\\_Theory/en](https://iccl.inf.tu-dresden.de/web/Database_Theory/en)

# Review: Datalog

## A rule-based recursive query language

father(alice, bob)

mother(alice, carla)

Parent( $x, y$ )  $\leftarrow$  father( $x, y$ )

Parent( $x, y$ )  $\leftarrow$  mother( $x, y$ )

SameGeneration( $x, x$ )

SameGeneration( $x, y$ )  $\leftarrow$  Parent( $x, v$ )  $\wedge$  Parent( $y, w$ )  $\wedge$  SameGeneration( $v, w$ )

- Datalog is more complex than FO query answering
- Datalog is more expressive than FO query answering
- Semipositive Datalog with a successor ordering captures P
- Datalog containment is undecidable

Remaining question: How can Datalog query answering be implemented?

# Implementing Datalog

FO queries (and thus also CQs and UCQs) are supported by almost all DBMS

~> many specific implementation and optimisation techniques

How can Datalog queries be answered in practice?

~> techniques for dealing with recursion in DBMS query answering

# Implementing Datalog

FO queries (and thus also CQs and UCQs) are supported by almost all DBMS

~> many specific implementation and optimisation techniques

How can Datalog queries be answered in practice?

~> techniques for dealing with recursion in DBMS query answering

There are two major paradigms for answering recursive queries:

- **Bottom-up:** derive conclusions by applying rules to given facts
- **Top-down:** search for proofs to infer results given query

# Computing Datalog Query Answers Bottom-Up

We already saw a way to compute Datalog answers bottom-up:  
the step-wise computation of the consequence operator  $T_P$

Bottom-up computation is known under many names:

- **Forward-chaining** since rules are “chained” from premise to conclusion (common in logic programming)
- **Materialisation** since inferred facts are stored (“materialised”) (common in databases)
- **Saturation** since the input database is “saturated” with inferences (common in theorem proving)
- **Deductive closure** since we “close” the input under entailments (common in formal logic)

# Naive Evaluation of Datalog Queries

A direct approach for computing  $T_P^\infty$

```
01   $T_P^0 := \emptyset$ 
02   $i := 0$ 
03  repeat :
04       $T_P^{i+1} := \emptyset$ 
05      for  $H \leftarrow B_1 \wedge \dots \wedge B_\ell \in P$  :
06          for  $\theta \in B_1 \wedge \dots \wedge B_\ell(T_P^i)$  :
07               $T_P^{i+1} := T_P^{i+1} \cup \{H\theta\}$ 
08       $i := i + 1$ 
09  until  $T_P^{i-1} = T_P^i$ 
10  return  $T_P^i$ 
```

Notation for line 06/07:

- a substitution  $\theta$  is a mapping from variables to database elements
- for a formula  $F$ , we write  $F\theta$  for the formula obtained by replacing each free variable  $x$  in  $F$  by  $\theta(x)$
- for a CQ  $Q$  and database  $\mathcal{I}$ , we write  $\theta \in Q(\mathcal{I})$  if  $\mathcal{I} \models Q\theta$

# What's Wrong with Naive Evaluation?

An example Datalog program:

$e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5)$

(R1)  $T(x, y) \leftarrow e(x, y)$

(R2)  $T(x, z) \leftarrow T(x, y) \wedge T(y, z)$

$$T_P^0 = \emptyset$$

$$T_P^1 = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\}$$

$$T_P^2 = T_P^1 \cup \{T(1, 3), T(2, 4), T(3, 5)\}$$

$$T_P^3 = T_P^2 \cup \{T(1, 4), T(2, 5), T(1, 5)\}$$

$$T_P^4 = T_P^3 = T_P^\infty$$

# What's Wrong with Naive Evaluation?

An example Datalog program:

$$\begin{array}{l} e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \\ (R1) \quad T(x, y) \leftarrow e(x, y) \\ (R2) \quad T(x, z) \leftarrow T(x, y) \wedge T(y, z) \end{array}$$

How many body matches do we need to iterate over?

$$T_P^0 = \emptyset \quad \text{initialisation}$$

$$T_P^1 = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\}$$

$$T_P^2 = T_P^1 \cup \{T(1, 3), T(2, 4), T(3, 5)\}$$

$$T_P^3 = T_P^2 \cup \{T(1, 4), T(2, 5), T(1, 5)\}$$

$$T_P^4 = T_P^3 = T_P^\infty$$



# What's Wrong with Naive Evaluation?

An example Datalog program:

$$\begin{array}{l} e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \\ (R1) \quad T(x, y) \leftarrow e(x, y) \\ (R2) \quad T(x, z) \leftarrow T(x, y) \wedge T(y, z) \end{array}$$

How many body matches do we need to iterate over?

$$\begin{array}{ll} T_P^0 = \emptyset & \text{initialisation} \\ T_P^1 = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\} & 4 \text{ matches for } (R1) \\ T_P^2 = T_P^1 \cup \{T(1, 3), T(2, 4), T(3, 5)\} \\ T_P^3 = T_P^2 \cup \{T(1, 4), T(2, 5), T(1, 5)\} \\ T_P^4 = T_P^3 = T_P^\infty \end{array}$$

# What's Wrong with Naive Evaluation?

An example Datalog program:

$$\begin{array}{l} e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \\ (R1) \quad T(x, y) \leftarrow e(x, y) \\ (R2) \quad T(x, z) \leftarrow T(x, y) \wedge T(y, z) \end{array}$$

How many body matches do we need to iterate over?

$$\begin{array}{ll} T_P^0 = \emptyset & \text{initialisation} \\ T_P^1 = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\} & 4 \text{ matches for } (R1) \\ T_P^2 = T_P^1 \cup \{T(1, 3), T(2, 4), T(3, 5)\} & 4 \times (R1) + 3 \times (R2) \\ T_P^3 = T_P^2 \cup \{T(1, 4), T(2, 5), T(1, 5)\} & \\ T_P^4 = T_P^3 = T_P^\infty & \end{array}$$

# What's Wrong with Naive Evaluation?

An example Datalog program:

$$\begin{array}{l} e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \\ (R1) \quad T(x, y) \leftarrow e(x, y) \\ (R2) \quad T(x, z) \leftarrow T(x, y) \wedge T(y, z) \end{array}$$

How many body matches do we need to iterate over?

$$\begin{array}{ll} T_P^0 = \emptyset & \text{initialisation} \\ T_P^1 = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\} & 4 \text{ matches for } (R1) \\ T_P^2 = T_P^1 \cup \{T(1, 3), T(2, 4), T(3, 5)\} & 4 \times (R1) + 3 \times (R2) \\ T_P^3 = T_P^2 \cup \{T(1, 4), T(2, 5), T(1, 5)\} & 4 \times (R1) + 8 \times (R2) \\ T_P^4 = T_P^3 = T_P^\infty & \end{array}$$

# What's Wrong with Naive Evaluation?

An example Datalog program:

$$\begin{array}{l} e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \\ (R1) \quad T(x, y) \leftarrow e(x, y) \\ (R2) \quad T(x, z) \leftarrow T(x, y) \wedge T(y, z) \end{array}$$

How many body matches do we need to iterate over?

$T_P^0 = \emptyset$	initialisation
$T_P^1 = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\}$	4 matches for (R1)
$T_P^2 = T_P^1 \cup \{T(1, 3), T(2, 4), T(3, 5)\}$	$4 \times (R1) + 3 \times (R2)$
$T_P^3 = T_P^2 \cup \{T(1, 4), T(2, 5), T(1, 5)\}$	$4 \times (R1) + 8 \times (R2)$
$T_P^4 = T_P^3 = T_P^\infty$	$4 \times (R1) + 10 \times (R2)$

# What's Wrong with Naive Evaluation?

An example Datalog program:

$$\begin{array}{l} e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \\ (R1) \quad T(x, y) \leftarrow e(x, y) \\ (R2) \quad T(x, z) \leftarrow T(x, y) \wedge T(y, z) \end{array}$$

How many body matches do we need to iterate over?

$$\begin{array}{ll} T_P^0 = \emptyset & \text{initialisation} \\ T_P^1 = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\} & 4 \text{ matches for } (R1) \\ T_P^2 = T_P^1 \cup \{T(1, 3), T(2, 4), T(3, 5)\} & 4 \times (R1) + 3 \times (R2) \\ T_P^3 = T_P^2 \cup \{T(1, 4), T(2, 5), T(1, 5)\} & 4 \times (R1) + 8 \times (R2) \\ T_P^4 = T_P^3 = T_P^\infty & 4 \times (R1) + 10 \times (R2) \end{array}$$

In total, we considered 37 matches to derive 11 facts

# Less Naive Evaluation Strategies

Does it really matter how often we **consider** a rule match?  
After all, each fact is added only once . . .

# Less Naive Evaluation Strategies

Does it really matter how often we **consider** a rule match?

After all, each fact is added only once . . .

In practice, finding applicable rules takes significant time, even if the conclusion does not need to be added – iteration takes time!

~> **huge potential for optimisation**

## **Observation:**

we derive the same conclusions over and over again in each step

**Idea:** apply rules only to newly derived facts

~> **semi-naive evaluation**

# Semi-Naive Evaluation

The computation yields sets  $T_P^0 \subseteq T_P^1 \subseteq T_P^2 \subseteq \dots \subseteq T_P^\infty$

- For an IDB predicate R, let  $R^i$  be the “predicate” that contains exactly the R-facts in  $T_P^i$
- For  $i \leq 1$ , let  $\Delta_R^i$  be the collection of facts  $R^i \setminus R^{i-1}$

We can restrict rules to use only some computations.



# Semi-Naive Evaluation

The computation yields sets  $T_P^0 \subseteq T_P^1 \subseteq T_P^2 \subseteq \dots \subseteq T_P^\infty$

- For an IDB predicate R, let  $R^i$  be the “predicate” that contains exactly the R-facts in  $T_P^i$
- For  $i \leq 1$ , let  $\Delta_R^i$  be the collection of facts  $R^i \setminus R^{i-1}$

We can restrict rules to use only some computations.

**Some options for the computation in step  $i + 1$ :**

$T(x, z) \leftarrow T^i(x, y) \wedge T^i(y, z)$	same as original rule
$T(x, z) \leftarrow \Delta_T^i(x, y) \wedge \Delta_T^i(y, z)$	restrict to new facts
$T(x, z) \leftarrow \Delta_T^i(x, y) \wedge T^i(y, z)$	partially restrict to new facts
$T(x, z) \leftarrow T^i(x, y) \wedge \Delta_T^i(y, z)$	partially restrict to new facts

What to choose?

## Semi-Naive Evaluation (2)

Inferences that involve new and old facts are necessary:

$e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5)$

(R1)  $T(x, y) \leftarrow e(x, y)$

(R2)  $T(x, z) \leftarrow T(x, y) \wedge T(y, z)$

$$T_P^0 = \emptyset$$

$$\Delta_T^1 = \{T(1, 2), T(2, 3), T(3, 4), T(3, 4), T(4, 5)\} \quad T_P^1 = \Delta_T^1$$

$$\Delta_T^2 = \{T(1, 3), T(2, 4), T(3, 5)\} \quad T_P^2 = T_P^1 \cup \Delta_T^2$$

$$\Delta_T^3 = \{T(1, 4), T(2, 5), T(1, 5)\} \quad T_P^3 = T_P^2 \cup \Delta_T^3$$

$$\Delta_T^4 = \emptyset \quad T_P^4 = T_P^3 = T_P^\infty$$

## Semi-Naive Evaluation (2)

Inferences that involve new and old facts are necessary:

$$\begin{array}{l} e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \\ (R1) \quad T(x, y) \leftarrow e(x, y) \\ (R2) \quad T(x, z) \leftarrow T(x, y) \wedge T(y, z) \end{array}$$

$$\begin{array}{ll} \Delta_T^1 = \{T(1, 2), T(2, 3), T(3, 4), T(3, 4), T(4, 5)\} & T_P^0 = \emptyset \\ \Delta_T^2 = \{T(1, 3), T(2, 4), T(3, 5)\} & T_P^1 = \Delta_T^1 \\ \Delta_T^3 = \{T(1, 4), T(2, 5), T(1, 5)\} & T_P^2 = T_P^1 \cup \Delta_T^2 \\ \Delta_T^4 = \emptyset & T_P^3 = T_P^2 \cup \Delta_T^3 \\ & T_P^4 = T_P^3 = T_P^\infty \end{array}$$

To derive  $T(1, 4)$  in  $\Delta_T^3$ , we need to combine

$T(1, 3) \in \Delta_T^2$  with  $T(3, 4) \in \Delta_T^1$  or  $T(1, 2) \in \Delta_T^1$  with  $T(2, 4) \in \Delta_T^2$

## Semi-Naive Evaluation (2)

Inferences that involve new and old facts are necessary:

$$\begin{array}{l} e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \\ (R1) \quad T(x, y) \leftarrow e(x, y) \\ (R2) \quad T(x, z) \leftarrow T(x, y) \wedge T(y, z) \end{array}$$

$$\begin{array}{ll} \Delta_T^1 = \{T(1, 2), T(2, 3), T(3, 4), T(3, 4), T(4, 5)\} & T_P^0 = \emptyset \\ \Delta_T^2 = \{T(1, 3), T(2, 4), T(3, 5)\} & T_P^1 = \Delta_T^1 \\ \Delta_T^3 = \{T(1, 4), T(2, 5), T(1, 5)\} & T_P^2 = T_P^1 \cup \Delta_T^2 \\ \Delta_T^4 = \emptyset & T_P^3 = T_P^2 \cup \Delta_T^3 \\ & T_P^4 = T_P^3 = T_P^\infty \end{array}$$

To derive  $T(1, 4)$  in  $\Delta_T^3$ , we need to combine

$T(1, 3) \in \Delta_T^2$  with  $T(3, 4) \in \Delta_T^1$  or  $T(1, 2) \in \Delta_T^1$  with  $T(2, 4) \in \Delta_T^2$

$\leadsto$  rule  $T(x, z) \leftarrow \Delta_T^i(x, y) \wedge \Delta_T^i(y, z)$  is not enough

## Semi-Naive Evaluation (3)

**Correct approach:** consider only rule application that use **at least one** newly derived IDB atom

For example program:

$e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5)$

(R1)  $T(x, y) \leftarrow e(x, y)$

(R2.1)  $T(x, z) \leftarrow \Delta_T^i(x, y) \wedge T^i(y, z)$

(R2.2)  $T(x, z) \leftarrow T^i(x, y) \wedge \Delta_T^i(y, z)$

## Semi-Naive Evaluation (3)

**Correct approach:** consider only rule application that use **at least one** newly derived IDB atom

For example program:

$e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5)$

(R1)  $T(x, y) \leftarrow e(x, y)$

(R2.1)  $T(x, z) \leftarrow \Delta_T^i(x, y) \wedge T^i(y, z)$

(R2.2)  $T(x, z) \leftarrow T^i(x, y) \wedge \Delta_T^i(y, z)$

There is still redundancy here: the matches for  $T(x, z) \leftarrow \Delta_T^i(x, y) \wedge \Delta_T^i(y, z)$  are covered by both (R2.1) and (R2.2)

## Semi-Naive Evaluation (3)

**Correct approach:** consider only rule application that use **at least one** newly derived IDB atom

For example program:

$e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5)$

(R1)  $T(x, y) \leftarrow e(x, y)$

(R2.1)  $T(x, z) \leftarrow \Delta_T^i(x, y) \wedge T^i(y, z)$

(R2.2)  $T(x, z) \leftarrow T^i(x, y) \wedge \Delta_T^i(y, z)$

There is still redundancy here: the matches for  $T(x, z) \leftarrow \Delta_T^i(x, y) \wedge \Delta_T^i(y, z)$  are covered by both (R2.1) and (R2.2)

$\leadsto$  replace (R2.2) by the following rule:

(R2.2')  $T(x, z) \leftarrow T^{i-1}(x, y) \wedge \Delta_T^i(y, z)$

EDB atoms do not change, so their  $\Delta$  would be  $\emptyset$

$\leadsto$  ignore such rules after the first iteration

# Semi-Naive Evaluation: Example

$e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5)$

$$(R1) \quad T(x, y) \leftarrow e(x, y)$$

$$(R2.1) \quad T(x, z) \leftarrow \Delta_T^i(x, y) \wedge T^i(y, z)$$

$$(R2.2') \quad T(x, z) \leftarrow T^{i-1}(x, y) \wedge \Delta_T^i(y, z)$$

$$T_P^0 = \emptyset$$

$$T_P^1 = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\}$$

$$T_P^2 = T_P^1 \cup \{T(1, 3), T(2, 4), T(3, 5)\}$$

$$T_P^3 = T_P^2 \cup \{T(1, 4), T(2, 5), T(1, 5)\}$$

$$T_P^4 = T_P^3 = T_P^\infty$$



# Semi-Naive Evaluation: Example

$$\begin{array}{l} \text{e}(1, 2) \quad \text{e}(2, 3) \quad \text{e}(3, 4) \quad \text{e}(4, 5) \\ (R1) \quad \text{T}(x, y) \leftarrow \text{e}(x, y) \\ (R2.1) \quad \text{T}(x, z) \leftarrow \Delta_{\text{T}}^i(x, y) \wedge \text{T}^i(y, z) \\ (R2.2') \quad \text{T}(x, z) \leftarrow \text{T}^{i-1}(x, y) \wedge \Delta_{\text{T}}^i(y, z) \end{array}$$

How many body matches do we need to iterate over?

$$T_P^0 = \emptyset \quad \text{initialisation}$$

$$T_P^1 = \{\text{T}(1, 2), \text{T}(2, 3), \text{T}(3, 4), \text{T}(4, 5)\}$$

$$T_P^2 = T_P^1 \cup \{\text{T}(1, 3), \text{T}(2, 4), \text{T}(3, 5)\}$$

$$T_P^3 = T_P^2 \cup \{\text{T}(1, 4), \text{T}(2, 5), \text{T}(1, 5)\}$$

$$T_P^4 = T_P^3 = T_P^\infty$$

# Semi-Naive Evaluation: Example

$$\begin{array}{l} \text{e}(1, 2) \quad \text{e}(2, 3) \quad \text{e}(3, 4) \quad \text{e}(4, 5) \\ (R1) \quad \text{T}(x, y) \leftarrow \text{e}(x, y) \\ (R2.1) \quad \text{T}(x, z) \leftarrow \Delta_{\text{T}}^i(x, y) \wedge \text{T}^i(y, z) \\ (R2.2') \quad \text{T}(x, z) \leftarrow \text{T}^{i-1}(x, y) \wedge \Delta_{\text{T}}^i(y, z) \end{array}$$

How many body matches do we need to iterate over?

$$T_P^0 = \emptyset \quad \text{initialisation}$$

$$T_P^1 = \{\text{T}(1, 2), \text{T}(2, 3), \text{T}(3, 4), \text{T}(4, 5)\} \quad 4 \times (R1)$$

$$T_P^2 = T_P^1 \cup \{\text{T}(1, 3), \text{T}(2, 4), \text{T}(3, 5)\}$$

$$T_P^3 = T_P^2 \cup \{\text{T}(1, 4), \text{T}(2, 5), \text{T}(1, 5)\}$$

$$T_P^4 = T_P^3 = T_P^\infty$$

# Semi-Naive Evaluation: Example

$$\begin{array}{llll} & e(1, 2) & e(2, 3) & e(3, 4) & e(4, 5) \\ (R1) & T(x, y) & \leftarrow e(x, y) & & \\ (R2.1) & T(x, z) & \leftarrow \Delta_T^i(x, y) \wedge T^i(y, z) & & \\ (R2.2') & T(x, z) & \leftarrow T^{i-1}(x, y) \wedge \Delta_T^i(y, z) & & \end{array}$$

How many body matches do we need to iterate over?

$$T_P^0 = \emptyset \quad \text{initialisation}$$

$$T_P^1 = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\} \quad 4 \times (R1)$$

$$T_P^2 = T_P^1 \cup \{T(1, 3), T(2, 4), T(3, 5)\} \quad 3 \times (R2.1)$$

$$T_P^3 = T_P^2 \cup \{T(1, 4), T(2, 5), T(1, 5)\}$$

$$T_P^4 = T_P^3 = T_P^\infty$$

# Semi-Naive Evaluation: Example

$$\begin{array}{l} \text{e}(1, 2) \quad \text{e}(2, 3) \quad \text{e}(3, 4) \quad \text{e}(4, 5) \\ (R1) \quad T(x, y) \leftarrow \text{e}(x, y) \\ (R2.1) \quad T(x, z) \leftarrow \Delta_T^i(x, y) \wedge T^i(y, z) \\ (R2.2') \quad T(x, z) \leftarrow T^{i-1}(x, y) \wedge \Delta_T^i(y, z) \end{array}$$

How many body matches do we need to iterate over?

$$\begin{array}{ll} T_P^0 = \emptyset & \text{initialisation} \\ T_P^1 = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\} & 4 \times (R1) \\ T_P^2 = T_P^1 \cup \{T(1, 3), T(2, 4), T(3, 5)\} & 3 \times (R2.1) \\ T_P^3 = T_P^2 \cup \{T(1, 4), T(2, 5), T(1, 5)\} & 3 \times (R2.1), 2 \times (R2.2') \\ T_P^4 = T_P^3 = T_P^\infty & \end{array}$$

# Semi-Naive Evaluation: Example

$$\begin{array}{l} \text{e}(1, 2) \quad \text{e}(2, 3) \quad \text{e}(3, 4) \quad \text{e}(4, 5) \\ (R1) \quad \text{T}(x, y) \leftarrow \text{e}(x, y) \\ (R2.1) \quad \text{T}(x, z) \leftarrow \Delta_{\text{T}}^i(x, y) \wedge \text{T}^i(y, z) \\ (R2.2') \quad \text{T}(x, z) \leftarrow \text{T}^{i-1}(x, y) \wedge \Delta_{\text{T}}^i(y, z) \end{array}$$

How many body matches do we need to iterate over?

$$\begin{array}{ll} T_P^0 = \emptyset & \text{initialisation} \\ T_P^1 = \{\text{T}(1, 2), \text{T}(2, 3), \text{T}(3, 4), \text{T}(4, 5)\} & 4 \times (R1) \\ T_P^2 = T_P^1 \cup \{\text{T}(1, 3), \text{T}(2, 4), \text{T}(3, 5)\} & 3 \times (R2.1) \\ T_P^3 = T_P^2 \cup \{\text{T}(1, 4), \text{T}(2, 5), \text{T}(1, 5)\} & 3 \times (R2.1), 2 \times (R2.2') \\ T_P^4 = T_P^3 = T_P^\infty & 1 \times (R2.1), 1 \times (R2.2') \end{array}$$

# Semi-Naive Evaluation: Example

$$\begin{array}{l} \text{e}(1, 2) \quad \text{e}(2, 3) \quad \text{e}(3, 4) \quad \text{e}(4, 5) \\ (R1) \quad T(x, y) \leftarrow \text{e}(x, y) \\ (R2.1) \quad T(x, z) \leftarrow \Delta_T^i(x, y) \wedge T^i(y, z) \\ (R2.2') \quad T(x, z) \leftarrow T^{i-1}(x, y) \wedge \Delta_T^i(y, z) \end{array}$$

How many body matches do we need to iterate over?

$$\begin{array}{ll} T_P^0 = \emptyset & \text{initialisation} \\ T_P^1 = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\} & 4 \times (R1) \\ T_P^2 = T_P^1 \cup \{T(1, 3), T(2, 4), T(3, 5)\} & 3 \times (R2.1) \\ T_P^3 = T_P^2 \cup \{T(1, 4), T(2, 5), T(1, 5)\} & 3 \times (R2.1), 2 \times (R2.2') \\ T_P^4 = T_P^3 = T_P^\infty & 1 \times (R2.1), 1 \times (R2.2') \end{array}$$

In total, we considered 14 matches to derive 11 facts

# Semi-Naive Evaluation: Full Definition

In general, a rule of the form

$$H(\vec{x}) \leftarrow e_1(\vec{y}_1) \wedge \dots \wedge e_n(\vec{y}_n) \wedge I_1(\vec{z}_1) \wedge I_2(\vec{z}_2) \wedge \dots \wedge I_m(\vec{z}_m)$$

is transformed into  $m$  rules

$$H(\vec{x}) \leftarrow e_1(\vec{y}_1) \wedge \dots \wedge e_n(\vec{y}_n) \wedge \Delta_{I_1}^i(\vec{z}_1) \wedge I_2^i(\vec{z}_2) \wedge \dots \wedge I_m^i(\vec{z}_m)$$

$$H(\vec{x}) \leftarrow e_1(\vec{y}_1) \wedge \dots \wedge e_n(\vec{y}_n) \wedge I_1^{i-1}(\vec{z}_1) \wedge \Delta_{I_2}^i(\vec{z}_2) \wedge \dots \wedge I_m^i(\vec{z}_m)$$

...

$$H(\vec{x}) \leftarrow e_1(\vec{y}_1) \wedge \dots \wedge e_n(\vec{y}_n) \wedge I_1^{i-1}(\vec{z}_1) \wedge I_2^{i-1}(\vec{z}_2) \wedge \dots \wedge \Delta_{I_m}^i(\vec{z}_m)$$

## Advantages and disadvantages:

- Huge improvement over naive evaluation
- Some redundant computations remain (see example)
- Some overhead for implementation (store level of entailments)

# Summary and Outlook

Datalog queries can be evaluated bottom-up or top-down

Simplest practical bottom-up technique: semi-naive evaluation

**Next question:**

- Can we improve Datalog evaluation further?
- What about practical implementations?