Review: Datalog

A rule-based recursive query language

father(alice, bob)
mother(alice, carla)
Parent\((x, y)\) ← father\((x, y)\)
Parent\((x, y)\) ← mother\((x, y)\)
SameGeneration\((x, x)\)
SameGeneration\((x, y)\) ← Parent\((x, v)\) ∧ Parent\((y, w)\) ∧ 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 DBMSs, so many specific implementation and optimisation techniques are used.

How can Datalog queries be answered in practice?
Many techniques for dealing with recursion in DBMS query answering are used.

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 $T_P^\infty$

```plaintext
01 $T_P^0 := \emptyset$
02 $i := 0$
03 repeat :
04 $T_P^{i+1} := \emptyset$
05 for $H \leftarrow B_1 \land \ldots \land B_\ell \in P$
06 for $\theta \in B_1 \land \ldots \land B_\ell(T_P^i)$:
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 $I$, we write $\theta \in Q(I)$ if $I \models Q_\theta$

Markus Krötzsch, 4th June 2019
What’s Wrong with Naive Evaluation?

An example Datalog program:

\[ e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \]

\[ (R1) \quad T(x, y) \leftarrow e(x, y) \]

\[ (R2) \quad T(x, z) \leftarrow T(x, y) \land T(y, z) \]

\[
T_P^0 = \emptyset \\
T_P^1 = \{ T(1, 2), T(2, 3), T(3, 4), T(4, 5) \} \\
T_P^2 = T_P^1 \cup \{ T(1, 3), T(2, 4), T(3, 5) \} \\
T_P^3 = T_P^2 \cup \{ T(1, 4), T(2, 5), T(1, 5) \} \\
T_P^4 = T_P^3 = T_P^\infty
\]
What’s Wrong with Naive Evaluation?

An example Datalog program:

\[
e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5)
\]

\[
(R1) \quad T(x, y) \leftarrow e(x, y)
\]

\[
(R2) \quad T(x, z) \leftarrow T(x, y) \land T(y, z)
\]

How many body matches do we need to iterate over?

\[
T^0_P = \emptyset \quad \text{initialisation}
\]

\[
T^1_P = \{ T(1, 2), T(2, 3), T(3, 4), T(4, 5) \}
\]

\[
T^2_P = T^1_P \cup \{ T(1, 3), T(2, 4), T(3, 5) \}
\]

\[
T^3_P = T^2_P \cup \{ T(1, 4), T(2, 5), T(1, 5) \}
\]

\[
T^4_P = T^3_P = T^\infty_P
\]

In total, we considered 37 matches to derive 11 facts.
What’s Wrong with Naive Evaluation?

An example Datalog program:

\[
\begin{align*}
&\text{e}(1, 2) \quad \text{e}(2, 3) \quad \text{e}(3, 4) \quad \text{e}(4, 5) \\
&(R1) \quad T(x, y) \leftarrow \text{e}(x, y) \\
&(R2) \quad T(x, z) \leftarrow T(x, y) \land T(y, z)
\end{align*}
\]

How many body matches do we need to iterate over?

\[
\begin{align*}
&T^0_P = \emptyset \quad \text{initialisation} \\
&T^1_P = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\} \quad \text{4 matches for } (R1) \\
&T^2_P = T^1_P \cup \{T(1, 3), T(2, 4), T(3, 5)\} \\
&T^3_P = T^2_P \cup \{T(1, 4), T(2, 5), T(1, 5)\} \\
&T^4_P = T^3_P = T^\infty
\end{align*}
\]

In total, we considered 37 matches to derive 11 facts.
What’s Wrong with Naive Evaluation?

An example Datalog program:

\[
\begin{align*}
\text{e(1, 2)} & \quad \text{e(2, 3)} & \quad \text{e(3, 4)} & \quad \text{e(4, 5)} \\
(R1) & \quad \text{T(x, y)} \leftarrow \text{e(x, y)} \\
(R2) & \quad \text{T(x, z)} \leftarrow \text{T(x, y)} \land \text{T(y, z)}
\end{align*}
\]

How many body matches do we need to iterate over?

\[
\begin{align*}
T^0_P &= \emptyset & \text{initialisation} \\
T^1_P &= \{\text{T(1, 2), T(2, 3), T(3, 4), T(4, 5)}\} & 4 \text{ matches for (R1)} \\
T^2_P &= T^1_P \cup \{\text{T(1, 3), T(2, 4), T(3, 5)}\} & 4 \times (R1) + 3 \times (R2) \\
T^3_P &= T^2_P \cup \{\text{T(1, 4), T(2, 5), T(1, 5)}\} \\
T^4_P &= T^3_P = T^\infty \\
\end{align*}
\]

In total, we considered 37 matches to derive 11 facts.
What’s Wrong with Naive Evaluation?

An example Datalog program:

\[
\begin{align*}
&\text{e}(1, 2) \quad \text{e}(2, 3) \quad \text{e}(3, 4) \quad \text{e}(4, 5) \\
&(R1) \quad \text{T}(x, y) \leftarrow \text{e}(x, y) \\
&(R2) \quad \text{T}(x, z) \leftarrow \text{T}(x, y) \land \text{T}(y, z)
\end{align*}
\]

How many body matches do we need to iterate over?

\[
\begin{align*}
&T^0_P = \emptyset \quad \text{initialisation} \\
&T^1_P = \{\text{T}(1, 2), \text{T}(2, 3), \text{T}(3, 4), \text{T}(4, 5)\} \quad 4 \text{ matches for } (R1) \\
&T^2_P = T^1_P \cup \{\text{T}(1, 3), \text{T}(2, 4), \text{T}(3, 5)\} \quad 4 \times (R1) + 3 \times (R2) \\
&T^3_P = T^2_P \cup \{\text{T}(1, 4), \text{T}(2, 5), \text{T}(1, 5)\} \quad 4 \times (R1) + 8 \times (R2) \\
&T^4_P = T^3_P = T^\infty_P
\end{align*}
\]
What’s Wrong with Naive Evaluation?

An example Datalog program:

\begin{align*}
 & 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) \land T(y, z)
\end{align*}

How many body matches do we need to iterate over?

\begin{align*}
 T_P^0 & = \emptyset \quad \text{initialisation} \\
 T_P^1 & = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\} \quad 4 \text{ matches for } (R1) \\
 T_P^2 & = T_P^1 \cup \{T(1, 3), T(2, 4), T(3, 5)\} \quad 4 \times (R1) + 3 \times (R2) \\
 T_P^3 & = T_P^2 \cup \{T(1, 4), T(2, 5), T(1, 5)\} \quad 4 \times (R1) + 8 \times (R2) \\
 T_P^4 & = T_P^3 = T_P^\infty \quad 4 \times (R1) + 10 \times (R2)
\end{align*}
What’s Wrong with Naive Evaluation?

An example Datalog program:

\[
\begin{align*}
\text{e}(1, 2) & \quad \text{e}(2, 3) & \quad \text{e}(3, 4) & \quad \text{e}(4, 5) \\
(R1) & & & \\
\text{T}(x, y) & \leftarrow & \text{e}(x, y) \\
(R2) & & & \\
\text{T}(x, z) & \leftarrow & \text{T}(x, y) & \land & \text{T}(y, z)
\end{align*}
\]

How many body matches do we need to iterate over?

\[
\begin{align*}
T^0_P &= \emptyset & \text{initialisation} \\
T^1_P &= \{ \text{T}(1, 2), \text{T}(2, 3), \text{T}(3, 4), \text{T}(4, 5) \} & 4 \text{ matches for } (R1) \\
T^2_P &= T^1_P \cup \{ \text{T}(1, 3), \text{T}(2, 4), \text{T}(3, 5) \} & 4 \times (R1) + 3 \times (R2) \\
T^3_P &= T^2_P \cup \{ \text{T}(1, 4), \text{T}(2, 5), \text{T}(1, 5) \} & 4 \times (R1) + 8 \times (R2) \\
T^4_P &= T^3_P = T^\infty_P & 4 \times (R1) + 10 \times (R2)
\end{align*}
\]

In total, we considered 37 matches to derive 11 facts
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!

$\Rightarrow$ 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

$\Rightarrow$ semi-naive evaluation
The computation yields sets $T^0_P \subseteq T^1_P \subseteq T^2_P \subseteq \ldots \subseteq T^\infty_P$

- For an IDB predicate $R$, let $R^i$ be the “predicate” that contains exactly the $R$-facts in $T^i_P$
- For $i \leq 1$, let $\Delta^i_R$ 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 $T^0_p \subseteq T^1_p \subseteq T^2_p \subseteq \ldots \subseteq T^\infty_p$

- For an IDB predicate $R$, let $R^i$ be the “predicate” that contains exactly the $R$-facts in $T^i_p$
- For $i \leq 1$, let $\Delta^i_R$ 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 $i + 1$:**

- $T(x, z) \leftarrow T^i(x, y) \land T^i(y, z)$  
  \hspace{5em} same as original rule
- $T(x, z) \leftarrow \Delta^i_T(x, y) \land \Delta^i_T(y, z)$  
  \hspace{5em} restrict to new facts
- $T(x, z) \leftarrow \Delta^i_T(x, y) \land T^i(y, z)$  
  \hspace{5em} partially restrict to new facts
- $T(x, z) \leftarrow T^i(x, y) \land \Delta^i_T(y, z)$  
  \hspace{5em} partially restrict to new facts

**What to choose?**
Semi-Naive Evaluation (2)

Inferences that involve new and old facts are necessary:

\[ e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \]

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

\( (R2) \quad T(x, z) \leftarrow T(x, y) \land T(y, z) \)

\[ T_P^0 = \emptyset \]

\[ T_P^1 = \Delta_T^1 \]

\[ \Delta_T^1 = \{ T(1, 2), T(2, 3), T(3, 4), T(3, 4), T(4, 5) \} \]

\[ T_P^2 = T_P^1 \cup \Delta_T^2 \]

\[ \Delta_T^2 = \{ T(1, 3), T(2, 4), T(3, 5) \} \]

\[ T_P^3 = T_P^2 \cup \Delta_T^3 \]

\[ \Delta_T^3 = \{ T(1, 4), T(2, 5), T(1, 5) \} \]

\[ T_P^4 = T_P^3 = T_P^{\infty} \]

\[ \Delta_T^4 = \emptyset \]
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) \[ T(x, z) \leftarrow T(x, y) \land T(y, z) \]

\[ T^0_P = \emptyset \]

\[ \Delta^1_T = \{T(1, 2), T(2, 3), T(3, 4), T(3, 4), T(4, 5)\} \quad T^1_P = \Delta^1_T \]

\[ \Delta^2_T = \{T(1, 3), T(2, 4), T(3, 5)\} \quad T^2_P = T^1_P \cup \Delta^2_T \]

\[ \Delta^3_T = \{T(1, 4), T(2, 5), T(1, 5)\} \quad T^3_P = T^2_P \cup \Delta^3_T \]

\[ \Delta^4_T = \emptyset \quad T^4_P = T^3_P = T^\infty_P \]

To derive \( T(1, 4) \) in \( \Delta^3_T \), we need to combine
\( T(1, 3) \in \Delta^2_T \) with \( T(3, 4) \in \Delta^1_T \) or \( T(1, 2) \in \Delta^1_T \) with \( T(2, 4) \in \Delta^2_T \)
Semi-Naive Evaluation (2)

Inferences that involve new and old facts are necessary:

\[ e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \]

\[
(R1) \quad T(x, y) \leftarrow e(x, y)
\]

\[
(R2) \quad T(x, z) \leftarrow T(x, y) \land T(y, z)
\]

\[
\Delta^1_T = \{T(1, 2), T(2, 3), T(3, 4), T(3, 4), T(4, 5)\} \quad T_P^0 = \emptyset
\]

\[
\Delta^2_T = \{T(1, 3), T(2, 4), T(3, 5)\} \quad T_P^1 = \Delta^1_T
\]

\[
\Delta^3_T = \{T(1, 4), T(2, 5), T(1, 5)\} \quad T_P^2 = T_P^1 \cup \Delta^2_T
\]

\[
\Delta^4_T = \emptyset \quad T_P^3 = T_P^2 \cup \Delta^3_T
\]

\[
T_P^4 = T_P^3 = T_P^{\infty}
\]

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

\( T(1, 3) \in \Delta^2_T \) with \( T(3, 4) \in \Delta^1_T \) or \( T(1, 2) \in \Delta^1_T \) with \( T(2, 4) \in \Delta^2_T \)

\( \nrightarrow \) rule \( T(x, z) \leftarrow \Delta^1_T(x, y) \land \Delta^1_T(y, z) \) 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:

\[
\begin{align*}
&\text{e}(1, 2) \quad \text{e}(2, 3) \quad \text{e}(3, 4) \quad \text{e}(4, 5) \\
&(R1) \quad \text{T}(x, y) \leftarrow \text{e}(x, y) \\
&(R2.1) \quad \text{T}(x, z) \leftarrow \Delta_T^i(x, y) \land \text{T}^i(y, z) \\
&(R2.2) \quad \text{T}(x, z) \leftarrow \text{T}^i(x, y) \land \Delta_T^i(y, z)
\end{align*}
\]
Semi-Naive Evaluation (3)

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

For example program:

\[
\begin{align*}
\text{e(1, 2)} & \quad \text{e(2, 3)} & \quad \text{e(3, 4)} & \quad \text{e(4, 5)} \\
(R1) & \quad \text{T}(x, y) \leftarrow \text{e}(x, y) \\
(R2.1) & \quad \text{T}(x, z) \leftarrow \Delta^i_T(x, y) \land T^i(y, z) \\
(R2.2) & \quad \text{T}(x, z) \leftarrow T^i(x, y) \land \Delta^i_T(y, z)
\end{align*}
\]

There is still redundancy here: the matches for \(\text{T}(x, z) \leftarrow \Delta^i_T(x, y) \land \Delta^i_T(y, z)\) 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:

\[
\begin{align*}
e(1, 2) & \quad e(2, 3) & \quad e(3, 4) & \quad e(4, 5) \\
(R1) & \quad T(x, y) & \leftarrow & \quad e(x, y) \\
(R2.1) & \quad T(x, z) & \leftarrow & \quad \Delta_T^i(x, y) \land T^i(y, z) \\
(R2.2) & \quad T(x, z) & \leftarrow & \quad T^i(x, y) \land \Delta_T^i(y, z)
\end{align*}
\]

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

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

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

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

\(\leadsto\) ignore such rules after the first iteration
Semi-Naive Evaluation: Example

\[ e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \]

\[(R1) \quad T(x, y) \leftarrow e(x, y)\]

\[(R2.1) \quad T(x, z) \leftarrow \Delta^i_T(x, y) \land T^i(y, z)\]

\[(R2.2') \quad T(x, z) \leftarrow T^{i-1}(x, y) \land \Delta^i_T(y, z)\]

\[T^0_P = \emptyset\]
\[T^1_P = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\}\]
\[T^2_P = T^1_P \cup \{T(1, 3), T(2, 4), T(3, 5)\}\]
\[T^3_P = T^2_P \cup \{T(1, 4), T(2, 5), T(1, 5)\}\]
\[T^4_P = T^3_P = T^\infty_P\]

In total, we considered 14 matches to derive 11 facts.
Semi-Naive Evaluation: Example

\[ \text{e(1, 2)} \quad \text{e(2, 3)} \quad \text{e(3, 4)} \quad \text{e(4, 5)} \]

\[(R1) \quad T(x, y) \leftarrow e(x, y)\]

\[(R2.1) \quad T(x, z) \leftarrow \Delta^i_T(x, y) \land T^i(y, z)\]

\[(R2.2') \quad T(x, z) \leftarrow T^{i-1}(x, y) \land \Delta^i_T(y, z)\]

How many body matches do we need to iterate over?

\[ T^0_P = \emptyset \quad \text{initialisation} \]

\[ T^1_P = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\} \]

\[ T^2_P = T^1_P \cup \{T(1, 3), T(2, 4), T(3, 5)\} \]

\[ T^3_P = T^2_P \cup \{T(1, 4), T(2, 5), T(1, 5)\} \]

\[ T^4_P = T^3_P = T^\infty_P \]
Semi-Naive Evaluation: Example

\[
\begin{align*}
\text{e(1, 2)} & \quad \text{e(2, 3)} & \quad \text{e(3, 4)} & \quad \text{e(4, 5)} \\
\text{(R1)} & \quad \text{T}(x, y) \leftarrow \text{e}(x, y) \\
\text{(R2.1)} & \quad \text{T}(x, z) \leftarrow \Delta^i_T(x, y) \land T^i(y, z) \\
\text{(R2.2')} & \quad \text{T}(x, z) \leftarrow T^{i-1}(x, y) \land \Delta^i_T(y, z)
\end{align*}
\]

How many body matches do we need to iterate over?

\[
\begin{align*}
T^0_P & = \emptyset & \text{initialisation} \\
T^1_P & = \{\text{T}(1, 2), \text{T}(2, 3), \text{T}(3, 4), \text{T}(4, 5)\} & 4 \times (R1) \\
T^2_P & = T^1_P \cup \{\text{T}(1, 3), \text{T}(2, 4), \text{T}(3, 5)\} \\
T^3_P & = T^2_P \cup \{\text{T}(1, 4), \text{T}(2, 5), \text{T}(1, 5)\} \\
T^4_P & = T^3_P = T^\infty
\end{align*}
\]

In total, we considered 14 matches to derive 11 facts.
Semi-Naive Evaluation: Example

\( \text{e}(1, 2) \quad \text{e}(2, 3) \quad \text{e}(3, 4) \quad \text{e}(4, 5) \)

\((R1)\)  \( T(x, y) \leftarrow \text{e}(x, y) \)

\((R2.1)\)  \( T(x, z) \leftarrow \Delta^i_T(x, y) \land T^i(y, z) \)

\((R2.2')\)  \( T(x, z) \leftarrow T^{i-1}(x, y) \land \Delta^i_T(y, z) \)

How many body matches do we need to iterate over?

\[ T^0_P = \emptyset \quad \text{initialisation} \]

\[ T^1_P = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\} \quad 4 \times (R1) \]

\[ T^2_P = T^1_P \cup \{T(1, 3), T(2, 4), T(3, 5)\} \quad 3 \times (R2.1) \]

\[ T^3_P = T^2_P \cup \{T(1, 4), T(2, 5), T(1, 5)\} \]

\[ T^4_P = T^3_P = T^\infty_P \]
Semi-Naive Evaluation: Example

\[ e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \]

\[(R1) \quad T(x, y) \leftarrow e(x, y)\]

\[(R2.1) \quad T(x, z) \leftarrow \Delta_T^i(x, y) \land T^i(y, z)\]

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

How many body matches do we need to iterate over?

\[ T_P^0 = \emptyset \quad \text{initialisation} \]

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

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

\[ T_P^3 = T_P^2 \cup \{ T(1, 4), T(2, 5), T(1, 5) \} \quad 3 \times (R2.1), 2 \times (R2.2') \]

\[ T_P^4 = T_P^3 = T_P^\infty \]
Semi-Naive Evaluation: Example

\[ e(1, 2) \quad e(2, 3) \quad e(3, 4) \quad e(4, 5) \]

\[ (R1) \quad T(x, y) \leftarrow e(x, y) \]

\[ (R2.1) \quad T(x, z) \leftarrow \Delta^i_T(x, y) \land T^i(y, z) \]

\[ (R2.2') \quad T(x, z) \leftarrow T^{i-1}(x, y) \land \Delta^i_T(y, z) \]

How many body matches do we need to iterate over?

\[ T^0_P = \emptyset \quad \text{initialisation} \]

\[ T^1_P = \{T(1, 2), T(2, 3), T(3, 4), T(4, 5)\} \quad 4 \times (R1) \]

\[ T^2_P = T^1_P \cup \{T(1, 3), T(2, 4), T(3, 5)\} \quad 3 \times (R2.1) \]

\[ T^3_P = T^2_P \cup \{T(1, 4), T(2, 5), T(1, 5)\} \quad 3 \times (R2.1), 2 \times (R2.2') \]

\[ T^4_P = T^3_P = T_P^\infty \quad 1 \times (R2.1), 1 \times (R2.2') \]
Semi-Naive Evaluation: Example

\begin{align*}
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 T^i_+(x,y) \land T^i(y,z) \\
(R2.2') & \quad T(x,z) \leftarrow T^{i-1}(x,y) \land T^i_+(y,z)
\end{align*}

How many body matches do we need to iterate over?

\begin{align*}
T^0_P &= \emptyset & \text{initialisation} \\
T^1_P &= \{T(1,2), T(2,3), T(3,4), T(4,5)\} & 4 \times (R1) \\
T^2_P &= T^1_P \cup \{T(1,3), T(2,4), T(3,5)\} & 3 \times (R2.1) \\
T^3_P &= T^2_P \cup \{T(1,4), T(2,5), T(1,5)\} & 3 \times (R2.1), 2 \times (R2.2') \\
T^4_P &= T^3_P = T^\infty_P & 1 \times (R2.1), 1 \times (R2.2')
\end{align*}

In total, we considered 14 matches to derive 11 facts
Semi-Naive Evaluation: Full Definition

In general, a rule of the form

\[ H(\vec{x}) \leftarrow e_1(\vec{y}_1) \land \ldots \land e_n(\vec{y}_n) \land I_1(\vec{z}_1) \land I_2(\vec{z}_2) \land \ldots \land I_m(\vec{z}_m) \]

is transformed into \( m \) rules

\[ H(\vec{x}) \leftarrow e_1(\vec{y}_1) \land \ldots \land e_n(\vec{y}_n) \land \Delta_{i_1}(\vec{z}_1) \land I_2(\vec{z}_2) \land \ldots \land I_m(\vec{z}_m) \]
\[ H(\vec{x}) \leftarrow e_1(\vec{y}_1) \land \ldots \land e_n(\vec{y}_n) \land I_{i_1}(\vec{z}_1) \land \Delta_{i_2}(\vec{z}_2) \land \ldots \land I_{i_m}(\vec{z}_m) \]
\[ \ldots \]
\[ H(\vec{x}) \leftarrow e_1(\vec{y}_1) \land \ldots \land e_n(\vec{y}_n) \land I_{i_1}(\vec{z}_1) \land I_{i_2}(\vec{z}_2) \land \ldots \land \Delta_{i_m}(\vec{z}_m) \]

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?