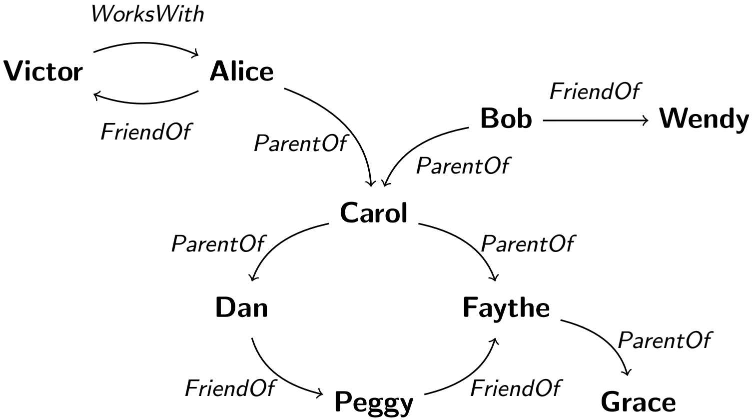
From relation algebra to semi-join algebra: an approach for graph query optimization

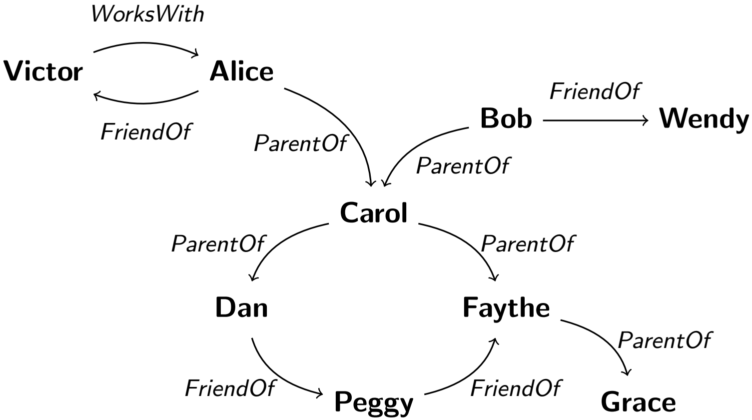
Jelle Hellings[1]
Catherine L. Pilachowski[2] Dirk Van Gucht[2]
Marc Gyssens[1] Yuqing Wu[3]

[1] Hasselt University
[2] Indiana University
[3] Pomona College

Graph queries: data model

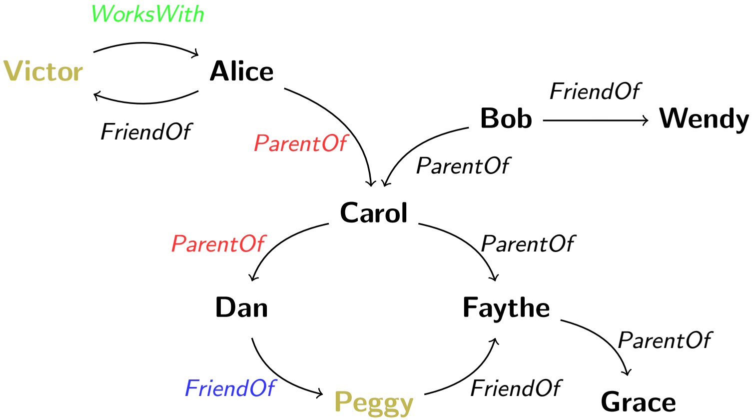


Graph queries: basic path queries



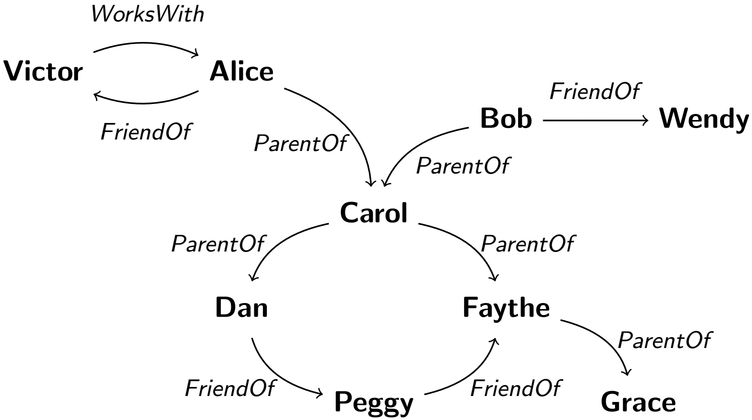
$$(\mathit{WorksWith} \cup \mathit{FriendOf}) \circ [\mathit{ParentOf}]^{+} \circ \mathit{FriendOf}$$

Graph queries: basic path queries



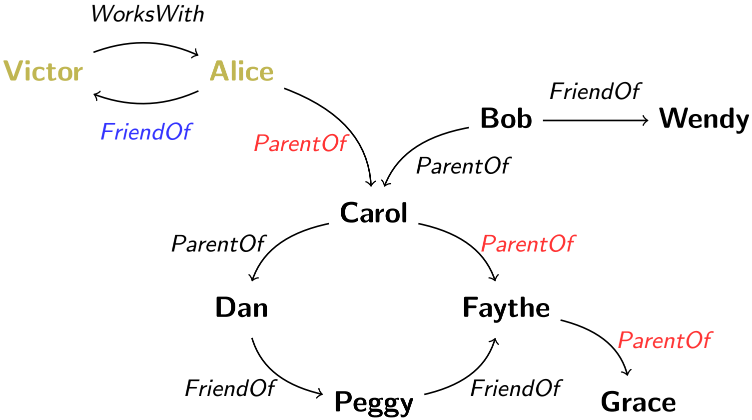
$$(\mathit{WorksWith} \cup \mathit{FriendOf}) \circ [\mathit{ParentOf}]^{+} \circ \mathit{FriendOf}$$

Graph queries: node-tests and branching



$$\pi_1[\mathit{ParentOf} \circ \mathit{ParentOf} \circ \mathit{ParentOf}] \circ \mathit{FriendOf}$$

Graph queries: node-tests and branching



$$\pi_1[\mathit{ParentOf} \circ \mathit{ParentOf} \circ \mathit{ParentOf}] \circ \mathit{FriendOf}$$

Graph querying: relation algebra

<table>
<tr><th>id</th><th>$\cup$</th><th>$\circ$</th><th>$+$</th><th>$\frown$</th><th>π</th><th>$\overline{\pi}$</th><th>$\cap$</th><th>$-$</th><th>di</th></tr>
<tr><td colspan="4">RPQs</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="5">2RPQs</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="6">Nested RPQs</td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="9">Navigational XPath, Graph XPath</td><td></td></tr>
<tr><td colspan="10">FO[3] $+$ transitive closure</td></tr>
</table>

Relation algebra and query evaluation

id	$\cup$	$\circ$	$+$	$\frown$	π	$\overline{\pi}$	$\cap$	$-$	di

Cheap ($\cup$, $\frown$, π, $\cap$, $-$).
Cost linearly upper bounded by operands

In between (id, $\overline{\pi}$).
Cost linearly upper bounded by #nodes

Expensive ($\circ$, $+$, di).
Worst-case quadratically lower bounded by #nodes

Naive query evaluation: an inefficient example

Return pairs of (great-grandparent, friend)

$$\pi_1[\mathit{ParentOf} \circ \mathit{ParentOf} \circ \mathit{ParentOf}] \circ \mathit{FriendOf}$$

1. Compute (grandparent, grandchild):
   $X = \mathit{ParentOf} \circ \mathit{ParentOf}$
2. Compute (great-grandparent, great-grandchild):
   $Y = \mathit{ParentOf} \circ X$
3. Throw away the great-grandchildren:
   $Z = \pi_1[Y]$
4. Compute (great-grandparent, friend):
   $\text{Result} = Z \circ \mathit{FriendOf}$

Optimize query evaluation: add specialized operators?

Return pairs of (great-grandparent, friend)

$$\pi_1[\mathit{ParentOf} \circ \mathit{ParentOf} \circ \mathit{ParentOf}] \circ \mathit{FriendOf}$$

1. Compute (grandparent, ???):
   $X = \mathit{ParentOf} \ltimes \mathit{ParentOf}$
2. Compute (great-grandparent, ???):
   $Y = \mathit{ParentOf} \ltimes (X)$
3. Throw away ???:
   $Z = \pi_1[Y]$
4. Compute (great-grandparent, friend):
   $\text{Result} = Z \rtimes \mathit{FriendOf}$

$$\pi_1[\mathit{ParentOf} \ltimes (\mathit{ParentOf} \ltimes \mathit{ParentOf})] \rtimes \mathit{FriendOf}$$

Simple idea: automatic query rewriting

- Rewrite composition into semi-joins
- Rewrite transitive closure into fixpoints

In such a way that the rewritten query is equivalent

When are expressions equivalent?

Definition

Queries q_1 and q_2 are

path-equivalent if, for every graph $\mathcal{G}$, $[\![q_1]\!]_\mathcal{G} = [\![q_2]\!]_\mathcal{G}$
(denoted by $q_1 \equiv_{\text{path}} q_2$)

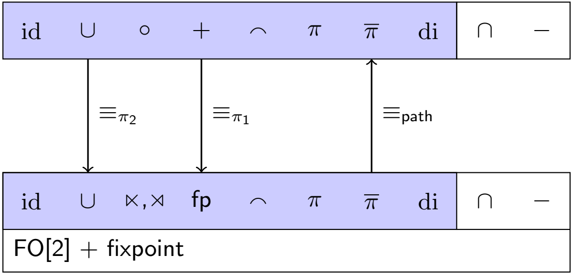
left-projection-equivalent if, for every graph $\mathcal{G}$, $[\![q_1]\!]_\mathcal{G}|_1 = [\![q_2]\!]_\mathcal{G}|_1$
(denoted by $q_1 \equiv_{\pi_1} q_2$)

right-projection-equivalent if, for every graph $\mathcal{G}$, $[\![q_1]\!]_\mathcal{G}|_2 = [\![q_2]\!]_\mathcal{G}|_2$
(denoted by $q_1 \equiv_{\pi_2} q_2$)

Example

- $R \cap S \equiv_{\text{path}} R - (R - S)$
- $R \circ S \equiv_{\pi_1} R \ltimes S$
- $\pi_1[R \circ S] \equiv_{\text{path}} \pi_1[R \ltimes S]$

The main result



- Collapse also holds for fragments (that include π)
- Example: Nested RPQs are projection-equivalent to expressions using only id, $\cup$, $\ltimes$, $\rtimes$, fp, $\frown$, and π

Intersection $\cap$ and difference $-$

Issues when combining composition with $\cap$ or $-$

$$(\mathit{FriendOf} \circ \mathit{FriendOf}) \cap \mathit{FriendOf}$$

- *Restricting*: use $\cap$ and $-$ only on composition-free expressions
  - Exact syntactic fragment of FO[3] $+$ *TC* that is projection-equivalent to FO[2] $+$ *fixpoint*.
- *Data models*: usage of $\cap$ and $-$ is sometimes redundant
  - Sibling-ordered trees: $\mathrm{FO}^{\mathrm{tree}} \preceq_\pi$ FO[2] $+$ fixpoints.
  - Downward queries on trees [DBPL 2015]
  - ...
- *Partial rewriting*: keep compositions when necessary

The rewrite functions - partial rewriting

$$\tau(e) \equiv_{\mathsf{path}} e \qquad \tau_{\pi_1}(e) \equiv_{\pi_1} e \qquad \tau_{\pi_2}(e) \equiv_{\pi_2} e$$

$$\tau_{\circ_1}(e;\varepsilon) \equiv_{\pi_1} e \ltimes \varepsilon \qquad \tau_{\circ_2}(e;\varepsilon) \equiv_{\pi_2} \varepsilon \rtimes e$$

Example

$\pi_1[((\mathit{WorksOn} \circ \mathit{WorksOn}^{\frown}) \cap \mathit{FriendOf}) \circ \mathit{EditorOf}] \circ \mathit{StudentOf}$

The rewrite functions - partial rewriting

$$\tau(e) \equiv_{\mathsf{path}} e \qquad \tau_{\pi_1}(e) \equiv_{\pi_1} e \qquad \tau_{\pi_2}(e) \equiv_{\pi_2} e$$

$$\tau_{\circ_1}(e; \varepsilon) \equiv_{\pi_1} e \ltimes \varepsilon \qquad \tau_{\circ_2}(e; \varepsilon) \equiv_{\pi_2} \varepsilon \rtimes e$$

Example

$\pi_1[((\mathit{WorksOn} \circ \mathit{WorksOn}^{\frown}) \cap \mathit{FriendOf}) \circ \mathit{EditorOf}] \circ \mathit{StudentOf}$

$$\tau(e)$$

.

The rewrite functions - partial rewriting

$$\tau(e) \equiv_{\text{path}} e \qquad \tau_{\pi_1}(e) \equiv_{\pi_1} e \qquad \tau_{\pi_2}(e) \equiv_{\pi_2} e$$

$$\tau_{\circ_1}(e; \varepsilon) \equiv_{\pi_1} e \ltimes \varepsilon \qquad \tau_{\circ_2}(e; \varepsilon) \equiv_{\pi_2} \varepsilon \rtimes e$$

Example

$\pi_1[((\mathit{WorksOn} \circ \mathit{WorksOn}^{\frown}) \cap \mathit{FriendOf}) \circ \mathit{EditorOf}] \circ \mathit{StudentOf}$

$$\tau(e) = \tau_{\pi_2}(\pi_1[((W \circ W^{\frown}) \cap F) \circ E]) \rtimes \tau(S)$$

The rewrite functions - partial rewriting

$$\tau(e) \equiv_{\text{path}} e \qquad \tau_{\pi_1}(e) \equiv_{\pi_1} e \qquad \tau_{\pi_2}(e) \equiv_{\pi_2} e$$

$$\tau_{\circ_1}(e;\varepsilon) \equiv_{\pi_1} e \ltimes \varepsilon \qquad \tau_{\circ_2}(e;\varepsilon) \equiv_{\pi_2} \varepsilon \rtimes e$$

Example

$$\pi_1[((\mathit{WorksOn} \circ \mathit{WorksOn}^{\frown}) \cap \mathit{FriendOf}) \circ \mathit{EditorOf}] \circ \mathit{StudentOf}$$

$$\begin{aligned}\tau(e) &= \tau_{\pi_2}(\pi_1[((W \circ W^{\frown}) \cap F) \circ E]) \rtimes \tau(S)\\ &= \pi_1[\tau_{\pi_1}(((W \circ W^{\frown}) \cap F) \circ E)] \rtimes S\end{aligned}$$

.

The rewrite functions - partial rewriting

$$\tau(e) \equiv_{\mathsf{path}} e \qquad \tau_{\pi_1}(e) \equiv_{\pi_1} e \qquad \tau_{\pi_2}(e) \equiv_{\pi_2} e$$

$$\tau_{\circ_1}(e;\varepsilon) \equiv_{\pi_1} e \ltimes \varepsilon \qquad \tau_{\circ_2}(e;\varepsilon) \equiv_{\pi_2} \varepsilon \rtimes e$$

Example

$\pi_1[((\mathit{WorksOn} \circ \mathit{WorksOn}^{\frown}) \cap \mathit{FriendOf}) \circ \mathit{EditorOf}] \circ \mathit{StudentOf}$

$$\begin{aligned} \tau(e) &= \tau_{\pi_2}(\pi_1[((W \circ W^{\frown}) \cap F) \circ E]) \rtimes \tau(S) \\ &= \pi_1[\tau_{\pi_1}(((W \circ W^{\frown}) \cap F) \circ E)] \rtimes S \\ &= \pi_1[\tau_{\circ_1}((W \circ W^{\frown}) \cap F; \tau_{\pi_1}(E))] \rtimes S \end{aligned}$$

The rewrite functions - partial rewriting

$$\tau(e) \equiv_{\text{path}} e \qquad \tau_{\pi_1}(e) \equiv_{\pi_1} e \qquad \tau_{\pi_2}(e) \equiv_{\pi_2} e$$

$$\tau_{\circ_1}(e;\varepsilon) \equiv_{\pi_1} e \ltimes \varepsilon \qquad \tau_{\circ_2}(e;\varepsilon) \equiv_{\pi_2} \varepsilon \rtimes e$$

Example

$\pi_1[((\mathit{WorksOn} \circ \mathit{WorksOn}^{\frown}) \cap \mathit{FriendOf}) \circ \mathit{EditorOf}] \circ \mathit{StudentOf}$

$$\begin{aligned}\tau(e) &= \tau_{\pi_2}(\pi_1[((W \circ W^{\frown}) \cap F) \circ E]) \rtimes \tau(S)\\ &= \pi_1[\tau_{\pi_1}(((W \circ W^{\frown}) \cap F) \circ E)] \rtimes S\\ &= \pi_1[\tau_{\circ_1}((W \circ W^{\frown}) \cap F; \tau_{\pi_1}(E))] \rtimes S\\ &= \pi_1[(\tau(W \circ W^{\frown}) \cap \tau(F)) \ltimes E] \rtimes S\end{aligned}$$

.

The rewrite functions - partial rewriting

$$\tau(e) \equiv_{\mathsf{path}} e \qquad \tau_{\pi_1}(e) \equiv_{\pi_1} e \qquad \tau_{\pi_2}(e) \equiv_{\pi_2} e$$

$$\tau_{\circ_1}(e;\varepsilon) \equiv_{\pi_1} e \ltimes \varepsilon \qquad \tau_{\circ_2}(e;\varepsilon) \equiv_{\pi_2} \varepsilon \rtimes e$$

Example

$$\pi_1[((\mathit{WorksOn} \circ \mathit{WorksOn}^\frown) \cap \mathit{FriendOf}) \circ \mathit{EditorOf}] \circ \mathit{StudentOf}$$

$$\begin{aligned}
\tau(e) &= \tau_{\pi_2}(\pi_1[((W \circ W^\frown) \cap F) \circ E]) \rtimes \tau(S) \\
&= \pi_1[\tau_{\pi_1}(((W \circ W^\frown) \cap F) \circ E)] \rtimes S \\
&= \pi_1[\tau_{\circ_1}((W \circ W^\frown) \cap F; \tau_{\pi_1}(E))] \rtimes S \\
&= \pi_1[(\tau(W \circ W^\frown) \cap \tau(F)) \ltimes E] \rtimes S \\
&= \pi_1[((\tau(W) \circ \tau(W^\frown)) \cap F) \ltimes E] \rtimes S
\end{aligned}$$

.

The rewrite functions - partial rewriting

$$\tau(e) \equiv_{\text{path}} e \qquad \tau_{\pi_1}(e) \equiv_{\pi_1} e \qquad \tau_{\pi_2}(e) \equiv_{\pi_2} e$$

$$\tau_{\circ_1}(e;\varepsilon) \equiv_{\pi_1} e \ltimes \varepsilon \qquad \tau_{\circ_2}(e;\varepsilon) \equiv_{\pi_2} \varepsilon \rtimes e$$

Example

$\pi_1[((\mathit{WorksOn} \circ \mathit{WorksOn}^\frown) \cap \mathit{FriendOf}) \circ \mathit{EditorOf}] \circ \mathit{StudentOf}$

$$\begin{aligned}
\tau(e) &= \tau_{\pi_2}(\pi_1[((W \circ W^\frown) \cap F) \circ E]) \rtimes \tau(S) \\
&= \pi_1[\tau_{\pi_1}(((W \circ W^\frown) \cap F) \circ E)] \rtimes S \\
&= \pi_1[\tau_{\circ_1}((W \circ W^\frown) \cap F; \tau_{\pi_1}(E))] \rtimes S \\
&= \pi_1[(\tau(W \circ W^\frown) \cap \tau(F)) \ltimes E] \rtimes S \\
&= \pi_1[((\tau(W) \circ \tau(W^\frown)) \cap F) \ltimes E] \rtimes S \\
&= \pi_1[((W \circ W^\frown) \cap F) \ltimes E] \rtimes S.
\end{aligned}$$

Query optimization

- Cost of each operator
- Input size of each operator

- Number of necessary evaluation steps

Query optimization

- Cost of each operator ✓
- Input size of each operator

- Number of necessary evaluation steps

Query optimization

- Cost of each operator ✓
- Input size of each operator

Example

Let $R = \{(1, i) \mid 0 \leq i \leq m\}$. Consider

$$R \circ R^{\frown} \equiv_{\pi_1} R \ltimes R^{\frown}.$$

- Number of necessary evaluation steps

Query optimization

- Cost of each operator ✓
- Input size of each operator ✓

Example

Let $R = \{(1, i) \mid 0 \leq i \leq m\}$. Consider

$$R \circ R^{\frown} \equiv_{\pi_1} R \ltimes R^{\frown}.$$

Solution: use single-column evaluation algorithms

- Number of necessary evaluation steps

Query optimization

- Cost of each operator ✓
- Input size of each operator ✓

Example

Let $R = \{(1, i) \mid 0 \leq i \leq m\}$. Consider

$$R \circ R^{\frown} \equiv_{\pi_1} R \ltimes R^{\frown}.$$

Solution: use single-column evaluation algorithms

- Number of necessary evaluation steps ✗

Expressions and evaluation steps

Expression size we denote the *expression size* of e by $\|e\|$.

Evaluation size we denote the *evaluation size* of e by eval-steps(e).

Example

$$e_1 = ((R \circ R) \circ (R \circ R)) \circ ((R \circ R) \circ (R \circ R))$$
$$e_2 = R \ltimes (R \ltimes (R \ltimes (R \ltimes (R \ltimes (R \ltimes (R \ltimes R))))))$$

- $e_1 \equiv_{\pi_1} e_2$
- We have $\|e_1\| = 7$ and eval-steps$(e_1) = 3$:
  1. $X = R \circ R$
  2. $Y = X \circ X$
  3. Result $= Y \circ Y$
- We have $\|e_2\| = 7$ and eval-steps$(e_2) = 7$.

Evaluation size and unions

Example

$$e_1 = (A \cup B) \circ (C \cup D) \circ (E \cup F)$$
$$e_2 = A \ltimes (C \ltimes E) \cup A \ltimes (C \ltimes F) \cup \dots$$

- $e_1 \equiv_{\pi_1} e_2$
- We have $\|e_1\| = \text{eval-steps}(e_1) = 5$.
- We have $\|e_2\| = \text{eval-steps}(e_2) = 23$.

Evaluation size and unions

Example

$$e_1 = (A \cup B) \circ (C \cup D) \circ (E \cup F)$$
$$e_2 = A \ltimes (C \ltimes E) \cup A \ltimes (C \ltimes F) \cup \ldots$$

- $e_1 \equiv_{\pi_1} e_2$
- We have $\|e_1\| = \text{eval-steps}(e_1) = 5$.
- We have $\|e_2\| = \text{eval-steps}(e_2) = 23$.

$$e_3 = (A \ltimes X) \cup (B \ltimes X),$$
$$X = (C \ltimes Y) \cup (D \ltimes Y), Y = (E \cup F)$$

- $e_1 \equiv_{\pi_1} e_3'$
- We have $\|e_2'\| = 13$ and $\text{eval-steps}(e_2') = 7$.

Evaluation size and unions

Example

$$e_1 = (A \cup B) \circ (C \cup D) \circ (E \cup F)$$
$$e_2 = A \ltimes (C \ltimes E) \cup A \ltimes (C \ltimes F) \cup \ldots$$

- $e_1 \equiv_{\pi_1} e_2$
- We have $\|e_1\| = \text{eval-steps}(e_1) = 5$.
- We have $\|e_2\| = \text{eval-steps}(e_2) = 23$.

$$e_3 = (A \ltimes X) \cup (B \ltimes X),$$
$$X = (C \ltimes Y) \cup (D \ltimes Y), Y = (E \cup F)$$

- $e_1 \equiv_{\pi_1} e_3'$
- We have $\|e_2'\| = 13$ and $\text{eval-steps}(e_2') = 7$.
- $\tau_{\circ_i}(e; \varepsilon)$ does this! ✓

The main results (revised)

Theorem

Let e be an expression. We have $\tau(e) \equiv_{\text{path}} e$, $\tau_{\pi_i}(e) \equiv_{\pi_i} e$, and

1. $\text{eval-steps}(\tau(e)) \leq u + \|e\|$;
2. $\text{eval-steps}(\tau_{\pi_i}(e)) \leq u + \|e\|$;
3. $\|\tau(e)\| = \Theta(\|e\| \cdot 2^u)$ *in the worst case;*
4. $\|\tau_{\pi_i}(e)\| = \Theta(\|e\| \cdot 2^u)$ *in the worst case,*

with u the number of rewrite steps involving $\tau_{\circ_i}(e_1 \cup e_2; \varepsilon)$.

Conclusion and future work

1. Real-life systems
2. Relational databases
3. Intersection and difference elimination
4. Extending FO[3] (e.g. counting)

The FO[2] fixpoint we use

- Notation $\mathrm{fp}_{i,\mathfrak{N}}$[iterative case union base case]
- i specifies output column
- $\mathfrak{N}$ is a variable representing the growing output (node-test)
- Subset of traditional inflationary fixpoints

Example

The query $\pi_1[[\mathit{ParentOf}]^+ \circ \mathit{OwnsPet}]$ returns ancestors of pet-owners. We rewrite this into

$$\pi_1[\mathrm{fp}_{1,\mathfrak{N}}[\mathit{ParentOf} \ltimes \mathfrak{N} \text{ union } \mathit{ParentOf} \ltimes \mathit{OwnsPet}]]$$