Type Inference for Datalog with Complex Type Hierarchies

Max Schäfer Oege de Moor

semmle

Example Program

semmle/

```
reports_to(x,y) ← ∃ g.(in_group(x,g) ∧ manages(y,g))
                ∨ ∃ g,d.(manages(x,g) ∧ part_of(g,d)
                                      ∧ leads(y,d))
                ∨ senior_mgr(x) ∧ x≐y.
```

```
bonus(x,y)      ← ∃ s,f.(¬parttime(x) ∧ salary(x,s)
                         ∧ factor(x,f) ∧ y≐f*s)
                ∨ parttime(x) ∧ y≐50.0.

factor(x,f)     ← senior_mgr(x) ∧ f≐0.5
                ∨ ¬senior_mgr(x) ∧ f≐0.2.

query(x,y)      ← bonus(x,y) ∧ manager(x).
```

Example Program

semmle/

```
reports_to(x,y) ← ∃ g.(in_group(x,g) ∧ manages(y,g))
                ∨ ∃ g,d.(manages(x,g) ∧ part_of(g,d)
                                    ∧ leads(y,d))
                ∨ senior_mgr(x) ∧ x≐y.
```

```
bonus(x,y)      ← ∃ s,f.(¬parttime(x) ∧ salary(x,s)
                         ∧ factor(x,f) ∧ y≐f*s)
                ∨ parttime(x) ∧ y≐50.0.

factor(x,f)     ← senior_mgr(x) ∧ f≐0.5
                ∨ ¬senior_mgr(x) ∧ f≐0.2.

query(x,y)      ← bonus(x,y) ∧ manager(x).
```

Example Program

semmle/

```
reports_to(x,y) ← ∃ g.(in_group(x,g) ∧ manages(y,g))
                ∨ ∃ g,d.(manages(x,g) ∧ part_of(g,d)
                                      ∧ leads(y,d))
                ∨ senior_mgr(x) ∧ x≐y.
```

```
bonus(x,y)      ← ∃ s,f.(¬parttime(x) ∧ salary(x,s)
                         ∧ factor(x,f) ∧ float(y))
                ∨ parttime(x) ∧ float(y).

factor(x,f)     ← senior_mgr(x) ∧ float(f)
                ∨ ¬senior_mgr(x) ∧ float(f).

query(x,y)      ← bonus(x,y) ∧ manager(x).
```

Type Hierarchy and Schema (I)

semmle/

- Schema $\mathscr{S}$ assigns entity types to columns of extensionals

$$(\forall x, g.\mathtt{in\_group}(x, g) \rightarrow \mathtt{developer}(x) \wedge \mathtt{group}(g))$$

$$(\forall x, g.\mathtt{manages}(x, g) \rightarrow \mathtt{junior\_mgr}(x) \wedge \mathtt{group}(g))$$

$$(\forall x, s.\mathtt{salary}(x, s) \rightarrow \mathtt{employee}(x) \wedge \mathtt{float}(s))$$

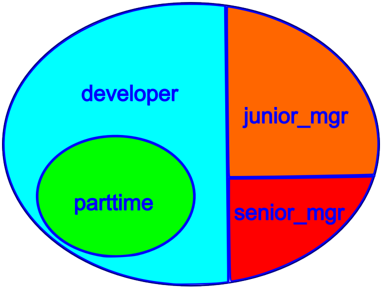
$$\ldots$$

Type Hierarchy and Schema (II)

semmle/

- Type Hierarchy $\mathscr{H}$ relates entity types

$$
\begin{aligned}
\forall x.\quad & (\mathtt{parttime}(x) \rightarrow \mathtt{developer}(x)) \\
\wedge\quad & (\mathtt{developer}(x) \rightarrow \mathtt{employee}(x)) \\
\wedge\quad & (\mathtt{manager}(x) \rightarrow \mathtt{employee}(x)) \\
\wedge\quad & (\mathtt{junior\_mgr}(x) \vee \mathtt{senior\_mgr}(x) \leftrightarrow \mathtt{manager}(x)) \\
\wedge\quad & \neg(\mathtt{developer}(x) \wedge \mathtt{manager}(x)) \\
\wedge\quad & \neg(\mathtt{junior\_mgr}(x) \wedge \mathtt{senior\_mgr}(x))
\end{aligned}
$$



Type-based Optimisation

semmle/

Example query:

```
bonus(x, y) ∧ manager(x)
```

Unfolding definitions:

```
(∃ s,f.(¬parttime(x) ∧ salary(x, s)
        ∧ factor(x,f) ∧ y≐f*s)
 ∨ parttime(x) ∧ y≐50.0)
∧ manager(x)
```

Type-based Optimisation

Example query:

$$\texttt{bonus(x, y)} \wedge \texttt{\textbf{manager}(x)}$$

Unfolding definitions:

$$\begin{array}{l}(\exists\ \texttt{s,f.}(\neg\texttt{\textbf{parttime}(x)} \wedge \texttt{\textbf{salary}(x, s)} \\ \qquad\qquad \wedge\ \texttt{factor(x,f)} \wedge \texttt{y}\dot{=}\texttt{f}\star\texttt{s}) \\ \ \cancel{\vee\ \texttt{\textbf{parttime}(x)} \wedge \texttt{y=50.0}}) \\ \wedge\ \texttt{\textbf{manager}(x)}\end{array}$$

Type specialisation:

$$\texttt{\textbf{parttime}}(x) \wedge \texttt{\textbf{manager}}(x) \models_{\mathscr{H}} \bot$$

Detecting Empty Programs

semmle/

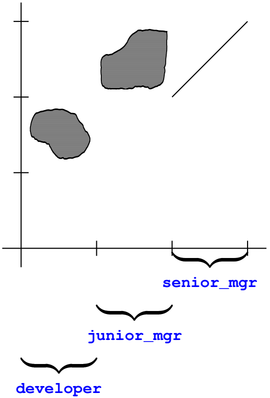
- want to know whether

$$\mathscr{H}, \mathscr{S}, \varphi \models \bot$$

- undecidable for arbitrary programs
- classic approach: find upper envelope

Cartesian Types

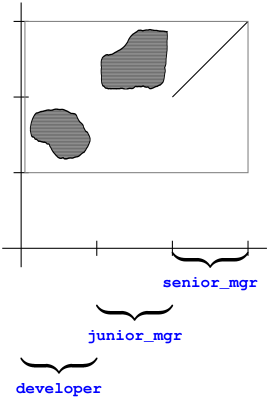
semmle/



$$
\begin{array}{l}
\texttt{reports\_to(x,y)} \leftarrow \\
\quad \exists\ \texttt{g.(}\mathbf{in\_group}\texttt{(x,g)} \wedge \mathbf{manages}\texttt{(y,g))} \\
\vee\ \exists\ \texttt{g,d.(}\mathbf{manages}\texttt{(x,g)} \wedge \mathbf{part\_of}\texttt{(g,d)} \\
\qquad\qquad\qquad\qquad \wedge\ \mathbf{leads}\texttt{(y,d))} \\
\vee\ \mathbf{senior\_mgr}\texttt{(x)} \wedge \texttt{x} \doteq \texttt{y.}
\end{array}
$$

Cartesian Types

semmle/



- one type per variable

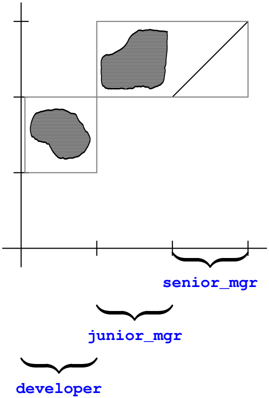
$$\lceil \texttt{reports\_to}(x, y) \rceil$$
$$= \quad \mathbf{employee}(x) \wedge \mathbf{manager}(y)$$

- erroneous query:

parttime(y) ∧ reports_to(x, y)

Disjunctive Types

semmle/



- disjunctive form

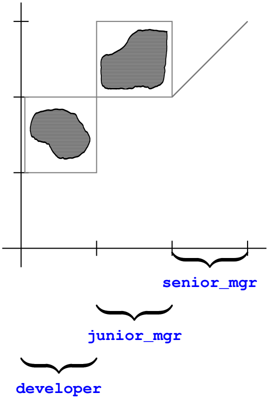
$$
\begin{aligned}
& \lceil \texttt{reports\_to}(x, y) \rceil \\
= \quad & \texttt{developer}(x) \wedge \texttt{junior\_mgr}(y) \\
\vee \quad & \texttt{manager}(x) \wedge \texttt{senior\_mgr}(y)
\end{aligned}
$$

- erroneous query:

```
parttime(x) ∧
reports_to(x, y) ∧ senior_mgr(y)
```

Disjunctive Types with Equality

semmle/



- disjunctive form

$$
\begin{aligned}
& \lceil \texttt{reports\_to}(x, y) \rceil \\
= \quad & \texttt{developer}(x) \land \texttt{junior\_mgr}(y) \\
\lor \quad & \texttt{junior\_mgr}(x) \land \texttt{senior\_mgr}(y) \\
\lor \quad & \texttt{senior\_mgr}(x) \land \texttt{senior\_mgr}(y) \\
& \qquad\qquad \land x = y
\end{aligned}
$$

Types are Programs

semmle/

$$
\begin{array}{lll}
\texttt{reports\_to(x,y)} & \leftarrow & \exists\ \texttt{g.}(\textbf{in\_group}\texttt{(x,g)} \land \textbf{manages}\texttt{(y,g))} \\
 & \lor & \exists\ \texttt{g,d.}(\textbf{manages}\texttt{(x,g)} \land \textbf{part\_of}\texttt{(g,d)} \\
 & & \qquad\qquad\qquad\qquad \land\ \textbf{leads}\texttt{(y,d))} \\
 & \lor & \textcolor{blue}{\textbf{senior\_mgr}}\texttt{(x)} \land \texttt{x} \doteq \texttt{y.}
\end{array}
$$

Types are Programs

semmle/

$$
\begin{array}{lll}
\texttt{reports\_to(x,y)} & \leftarrow & \exists\ \texttt{g.}(\textbf{in\_group}\texttt{(x,g)} \wedge \textbf{manages}\texttt{(y,g))} \\
 & \vee & \exists\ \texttt{g,d.}(\textbf{manages}\texttt{(x,g)} \wedge \textbf{part\_of}\texttt{(g,d)} \\
 & & \qquad\qquad\qquad\qquad \wedge\ \textbf{leads}\texttt{(y,d))} \\
 & \vee & \textbf{senior\_mgr}\texttt{(x)} \wedge \texttt{x} \doteq \texttt{y.}
\end{array}
$$

$$\Downarrow$$

$$
\begin{array}{lll}
\lceil\texttt{reports\_to(x,y)}\rceil & \leftarrow & \exists\ \texttt{g.}(\textbf{developer}\texttt{(x)} \wedge \textbf{group}\texttt{(g)} \\
 & & \qquad\qquad\qquad\quad \wedge\ \textbf{junior\_mgr}\texttt{(y))} \\
 & \vee & \exists\ \texttt{g,d.}(\textbf{junior\_mgr}\texttt{(x)} \wedge \textbf{group}\texttt{(g)} \\
 & & \qquad \wedge\ \textbf{department}\texttt{(d)} \wedge \textbf{senior\_mgr}\texttt{(y))} \\
 & \vee & \textbf{senior\_mgr}\texttt{(x)} \wedge \texttt{x} \doteq \texttt{y.}
\end{array}
$$

Types are Programs

semmle/

$$
\begin{aligned}
\mathtt{reports\_to(x,y)} \quad & \leftarrow \exists\ \mathtt{g.}(\mathbf{in\_group}\mathtt{(x,g)} \wedge \mathbf{manages}\mathtt{(y,g))} \\
& \vee \exists\ \mathtt{g,d.}(\mathbf{manages}\mathtt{(x,g)} \wedge \mathbf{part\_of}\mathtt{(g,d)} \\
& \qquad\qquad\qquad\qquad \wedge \mathbf{leads}\mathtt{(y,d))} \\
& \vee \mathbf{senior\_mgr}\mathtt{(x)} \wedge \mathtt{x} \doteq \mathtt{y.}
\end{aligned}
$$

$$\Downarrow$$

$$
\begin{aligned}
\lceil \mathtt{reports\_to(x,y)} \rceil & \leftarrow \mathbf{developer}\mathtt{(x)} \wedge \mathbf{junior\_mgr}\mathtt{(y)} \\
& \quad \wedge \exists\ \mathtt{g.}(\mathbf{group}\mathtt{(g))} \\
& \vee \mathbf{junior\_mgr}\mathtt{(x)} \wedge \mathbf{senior\_mgr}\mathtt{(y)} \\
& \quad \wedge \exists\ \mathtt{g.}(\mathbf{group}\mathtt{(g))} \wedge \exists\ \mathtt{d.}(\mathbf{department}\mathtt{(d))} \\
& \vee \mathbf{senior\_mgr}\mathtt{(x)} \wedge \mathtt{x} \doteq \mathtt{y.}
\end{aligned}
$$

Types as Upper Envelopes

- type inference: $\lceil \cdot \rceil : P \rightarrow T$ where
  - $T \subset P$
  - containment, emptiness decidable on T
  - $\mathscr{H}, \mathscr{S}, \varphi \models \lceil \varphi \rceil$

Typing Datalog Programs

- types are existential programs with monadic extensionals
  - polyadic intensionals
  - recursion, existentials, equality
  - no negation
- $\lceil e \rceil := \mathscr{S}(e)$ for extensionals
- $\lceil \neg\varphi \rceil := \top$

Main Results

semmle/

Soundness and Optimality

- for any $\varphi \in P$:
$$\mathscr{S}, \varphi \models \lceil \varphi \rceil$$
- for negation-free $\varphi \in P$ and $\vartheta \in T$:
$$\begin{array}{lll} \text{if} & \mathscr{S}, \mathscr{H}, \varphi & \models \vartheta \\ \text{then} & \mathscr{S}, \mathscr{H}, \lceil \varphi \rceil & \models \vartheta \end{array}$$

Representing Type Programs

semmle/

- intensional relations in T can always be eliminated:
- can be written as disjunction of conjunctions
- every conjunct of the form $t(x)$, $x \doteq y$ or $\exists z.t(z)$ for propositional t
  $\Rightarrow$ compact representation using BDDs
- containment and emptiness decidable

Conclusions

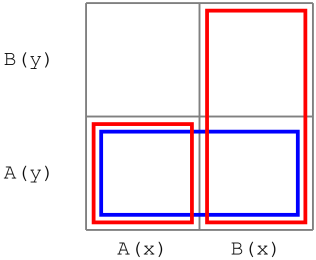
- natural, expressive language of types for Datalog
- support for complex type hierarchies
- simple, sound, optimal type inference
- compact representation of types

Containment Checking

semmle/

- $S <: T$ if every disjunct of S implies some disjunct of T
- obviously sound, $S <: T$ implies $S \models T$
- but incomplete!

Geometric Analysis

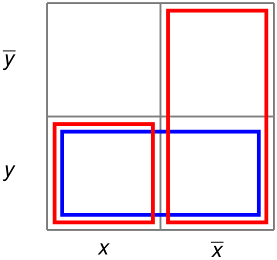


$$\underbrace{\overbrace{\mathtt{C}(x) \wedge \mathtt{A}(y)}^{\sigma_1}}_{S} \models \underbrace{\overbrace{\mathtt{A}(x) \wedge \mathtt{A}(y)}^{\tau_1} \vee \overbrace{\mathtt{B}(x) \wedge \mathtt{C}(y)}^{\tau_2}}_{T}$$

yet $\sigma_1 \not\models \tau_1$ and $\sigma_1 \not\models \tau_2$

Prime Implicants

semmle/



$$y \models x \wedge y \vee \neg x$$

- prime implicants of $x \wedge y \vee \neg x$: $\{y, \neg x\}$
- obtained by consensus formation: $(x \wedge y) \oplus_x \neg x = y$

Generalising to Types

semmle/

- new interpretation of consensus:

$$(x \wedge y) \oplus_x (\neg x \wedge (y \vee \neg y)) = (x \vee \neg x) \wedge y = y$$

  "disjunction on x, conjunction on y"

- can be generalised to types, need to do consensus on <u>sets</u> of variables
- $\texttt{sat}(T)$: all prime implicants of T

Complete Containment Check

If $S \models_{\mathscr{H}} T$, then $S <: \texttt{sat}(T)$.

Practical Feasibility