

Type Inference for Datalog with Complex Type Hierarchies

Max Schäfer Oege de Moor

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).
```

```
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).
```

```
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).
```

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

$(\forall x, g. \text{in_group}(x, g) \rightarrow \text{developer}(x) \wedge \text{group}(g))$

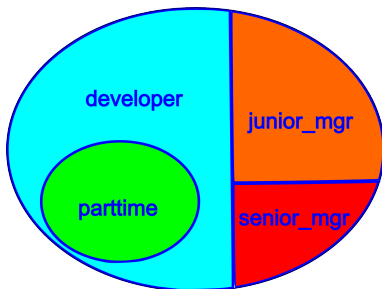
$(\forall x, g. \text{manages}(x, g) \rightarrow \text{junior_mgr}(x) \wedge \text{group}(g))$

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

...

- Type Hierarchy \mathcal{H} relates entity types

$\forall x.$ (parttime(x) \rightarrow developer(x))
 \wedge (developer(x) \rightarrow employee(x))
 \wedge (manager(x) \rightarrow employee(x))
 \wedge (junior_mgr(x) \vee senior_mgr(x) \leftrightarrow manager(x))
 \wedge \neg (developer(x) \wedge manager(x))
 \wedge \neg (junior_mgr(x) \wedge senior_mgr(x))



Example query:

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

Unfolding definitions:

$$\begin{aligned} &(\exists s, f. (\neg \text{parttime}(x) \wedge \text{salary}(x, s) \\ &\quad \wedge \text{factor}(x, f) \wedge y \doteq f * s) \\ &\vee \text{parttime}(x) \wedge y \doteq 50.0) \\ &\wedge \text{manager}(x) \end{aligned}$$

Example query:

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

Unfolding definitions:

$$\begin{aligned} & (\exists s, f. (\neg \text{parttime}(x) \wedge \text{salary}(x, s) \\ & \quad \wedge \text{factor}(x, f) \wedge y = f * s) \\ & \quad \wedge \text{parttime}(x) \wedge y = 50.0) \\ & \wedge \text{manager}(x) \end{aligned}$$

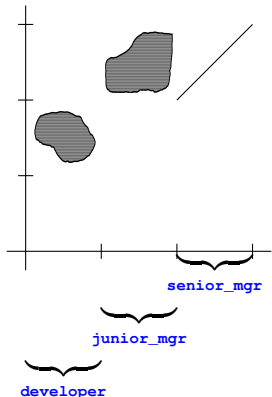
Type specialisation:

$$\text{parttime}(x) \wedge \text{manager}(x) \models_{\mathcal{H}} \perp$$

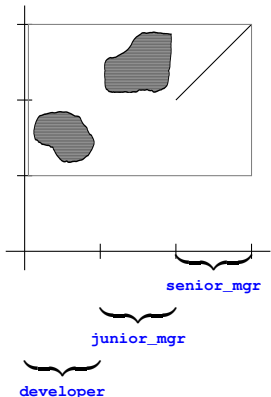
- want to know whether

$$\mathcal{H}, \mathcal{I}, \varphi \models \perp$$

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



```
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.
```

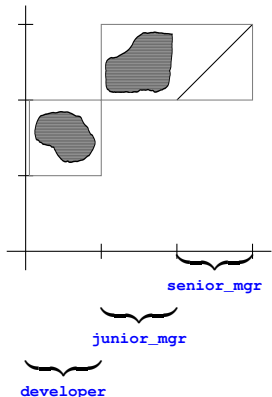


- one type per variable

$$\begin{aligned} & [\text{reports_to}(x, y)] \\ & = \text{employee}(x) \wedge \text{manager}(y) \end{aligned}$$

- erroneous query:

$$\text{parttime}(y) \wedge \text{reports_to}(x, y)$$

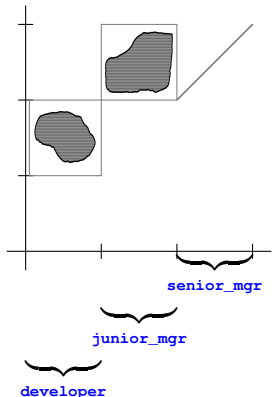


- disjunctive form

$$\begin{aligned} & [\text{reports_to}(x, y)] \\ &= \text{developer}(x) \wedge \text{junior_mgr}(y) \\ &\vee \text{manager}(x) \wedge \text{senior_mgr}(y) \end{aligned}$$

- erroneous query:

$$\begin{aligned} & \text{parttime}(x) \wedge \\ & \text{reports_to}(x, y) \wedge \text{senior_mgr}(y) \end{aligned}$$



- disjunctive form

$$\begin{aligned} & [\text{reports_to}(x, y)] \\ = & \text{developer}(x) \wedge \text{junior_mgr}(y) \\ \vee & \text{junior_mgr}(x) \wedge \text{senior_mgr}(y) \\ \vee & \text{senior_mgr}(x) \wedge \text{senior_mgr}(y) \\ & \wedge x = y \end{aligned}$$

`reports_to(x, y)` $\leftarrow \exists g. (\mathbf{in_group}(x, g) \wedge \mathbf{manages}(y, g))$
 $\vee \exists g, d. (\mathbf{manages}(x, g) \wedge \mathbf{part_of}(g, d)$
 $\wedge \mathbf{leads}(y, d))$
 $\vee \mathbf{senior_mgr}(x) \wedge x \dot{=} y.$

$$\begin{aligned} \text{reports_to}(x, y) &\leftarrow \exists g. (\mathbf{in_group}(x, g) \wedge \mathbf{manages}(y, g)) \\ &\vee \exists g, d. (\mathbf{manages}(x, g) \wedge \mathbf{part_of}(g, d) \\ &\quad \wedge \mathbf{leads}(y, d)) \\ &\vee \mathbf{senior_mgr}(x) \wedge x \dot{=} y. \end{aligned}$$

$$\begin{aligned} [\text{reports_to}(x, y)] &\leftarrow \exists g. (\mathbf{developer}(x) \wedge \mathbf{group}(g) \\ &\quad \wedge \mathbf{junior_mgr}(y)) \\ &\vee \exists g, d. (\mathbf{junior_mgr}(x) \wedge \mathbf{group}(g) \\ &\quad \wedge \mathbf{department}(d) \wedge \mathbf{senior_mgr}(y)) \\ &\vee \mathbf{senior_mgr}(x) \wedge x \dot{=} y. \end{aligned}$$

$$\begin{aligned} \text{reports_to}(x, y) &\leftarrow \exists g. (\mathbf{in_group}(x, g) \wedge \mathbf{manages}(y, g)) \\ &\vee \exists g, d. (\mathbf{manages}(x, g) \wedge \mathbf{part_of}(g, d) \\ &\quad \wedge \mathbf{leads}(y, d)) \\ &\vee \mathbf{senior_mgr}(x) \wedge x \dot{=} y. \end{aligned}$$

$$\begin{aligned} [\text{reports_to}(x, y)] &\leftarrow \mathbf{developer}(x) \wedge \mathbf{junior_mgr}(y) \\ &\quad \wedge \exists g. (\mathbf{group}(g)) \\ &\vee \mathbf{junior_mgr}(x) \wedge \mathbf{senior_mgr}(y) \\ &\quad \wedge \exists g. (\mathbf{group}(g)) \wedge \exists d. (\mathbf{department}(d)) \\ &\vee \mathbf{senior_mgr}(x) \wedge x \dot{=} y. \end{aligned}$$

- type inference: $[\cdot]: P \rightarrow T$ where
 - $T \subset P$
 - containment, emptiness decidable on T
 - $\mathcal{H}, \mathcal{S}, \varphi \models [\varphi]$

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

Soundness and Optimality

- for any $\varphi \in P$:

$$\mathcal{S}, \varphi \models [\varphi]$$

- for negation-free $\varphi \in P$ and $\vartheta \in T$:

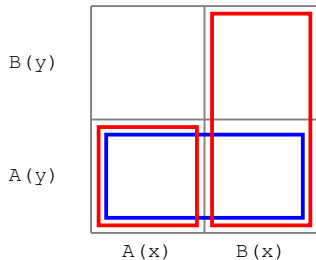
$$\text{if } \mathcal{S}, \mathcal{H}, \varphi \models \vartheta$$

$$\text{then } \mathcal{S}, \mathcal{H}, [\varphi] \models \vartheta$$

- 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

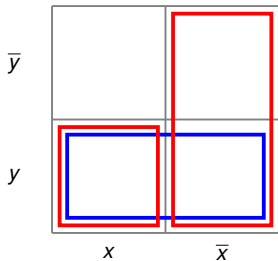
- natural, expressive language of types for Datalog
- support for complex type hierarchies
- simple, sound, optimal type inference
- compact representation of types

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



$$\underbrace{\overbrace{C(x) \wedge A(y)}^{\sigma_1}}_S \models \underbrace{\overbrace{A(x) \wedge A(y)}^{\tau_1} \vee \overbrace{B(x) \wedge C(y)}^{\tau_2}}_T$$

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



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

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

- 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 sets of variables
- $\text{sat}(T)$: all prime implicants of T

Complete Containment Check

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

