HUMBOLDT-UNIVERSITÄT ZU BERLIN

Learning Aggregate Queries Defined by First-Order Logic with Counting

Steffen van Bergerem and Nicole Schweikardt

ICDT 2025

How to be a Good Colleague

How to be a Good Colleague

Name	Popularity
Alice	5
Bob	1
Carol	2
Dan	3
Emma	1

(names changed for privacy reasons)

How to be a Good Colleague

Name	Popularity
Alice	5
Bob	1
Carol	2
Dan	3
Emma	1

(names changed for privacy reasons)

Name	Type of Cake
Alice	chocolate
Dan	lemon
Carol	strawberry
Alice	chocolate
Bob	carrot
Emma	apple
Dan	chocolate
Alice	strawberry
Carol	lemon

How to be a Good Colleague

Name	Popularity
Alice	5
Bob	1
Carol	2
Dan	3
Emma	1

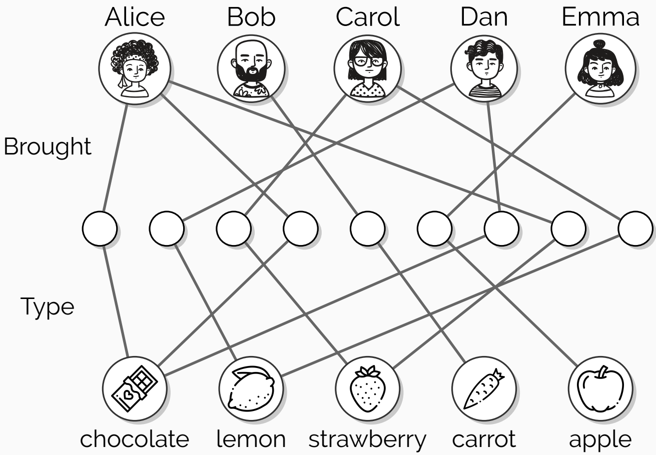
(names changed for privacy reasons)

Name	Type of Cake
Alice	chocolate
Dan	lemon
Carol	strawberry
Alice	chocolate
Bob	carrot
Emma	apple
Dan	chocolate
Alice	strawberry
Carol	lemon

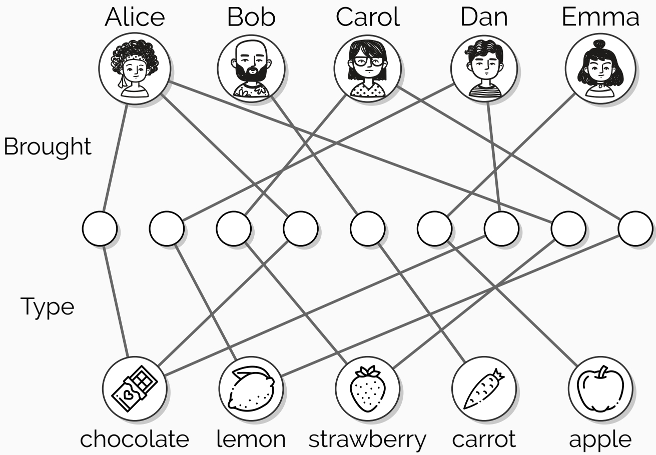
$$\text{Popularity} = 2 \cdot \#\text{chocolate cakes} + \#\text{other cakes}$$

How to be a Good Colleague

How to be a Good Colleague

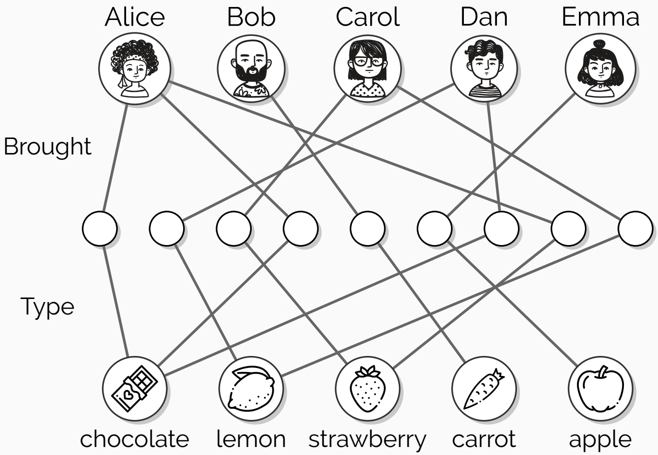


How to be a Good Colleague



(Alice, 5)
(Bob, 1)
(Carol, 2)
(Dan, 3)
(Emma, 1)

How to be a Good Colleague



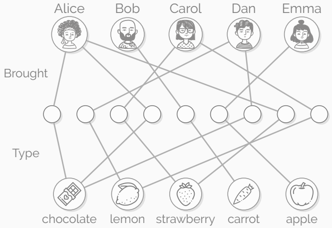
(Alice, 5)
(Bob, 1)
(Carol, 2)
(Dan, 3)
(Emma, 1)

$$p(x) = 2 \cdot \#(c).(Brought(x,c) \wedge Type(c, 🍫)) + \#(c).(Brought(x,c) \wedge \neg Type(c, 🍫))$$

Learning from Examples

Precomputation phase

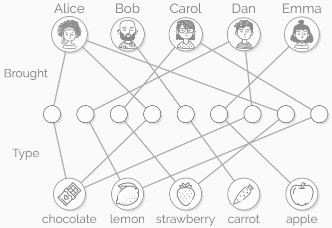
Given a database *D*, build index structure



Learning from Examples

Precomputation phase

Given a database D, build index structure



(Alice, 5)
(Bob, 1)
(Carol, 2)
(Dan, 3)
(Emma, 1)

Learning phase

Given list of labelled examples $(\bar{v}, \lambda) \in (U(D))^k \times \mathbb{Z}$

Return term $t(\bar{x}) \in \mathrm{FOC}_1$ (of certain maximum complexity)
such that $[\![t(\bar{v})]\!]^D = \lambda$ for all given examples $(\bar{v}, \lambda)$

or reject if there is no such term

Results on databases of polylogarithmic degree

Grohe and Ritzert, LICS 2017

Boolean-valued concepts definable in first-order logic can be learned in sublinear time.

v. B. and Schweikardt, CSL 2021

Boolean-valued concepts definable in first-order logic with counting or first-order logic with weight aggregation can be learned in sublinear time after quasi-linear-time precomputation.

Results on databases of polylogarithmic degree

Grohe and Ritzert, LICS 2017

Boolean-valued concepts definable in first-order logic can be learned in sublinear time.

v. B. and Schweikardt, CSL 2021

Boolean-valued concepts definable in first-order logic with counting or first-order logic with weight aggregation can be learned in sublinear time after quasi-linear-time precomputation.

v. B. and Schweikardt, ICDT 2025

Integer-valued concepts definable in first-order logic with counting can be learned in sublinear time after quasi-linear-time precomputation.

Main tool: locality results similar to Gaifman normal forms

First-Order Logic with Counting (FOC)

Counting terms

Formulas

First-Order Logic with Counting (FOC)

Counting terms

- $t_{🍫}(x) = \#(c).(Brought(x, c) \wedge Type(c, 🍫))$
- $t_{\neg 🍫}(x) = \#(c).(Brought(x, c) \wedge \neg Type(c, 🍫))$

Formulas

First-Order Logic with Counting (FOC)

Counting terms

- $t_{🍫}(x) = \#(c).(Brought(x, c) \wedge Type(c, 🍫))$
- $t_{\neg 🍫}(x) = \#(c).(Brought(x, c) \wedge \neg Type(c, 🍫))$
- $p(x) = 2 \cdot t_{🍫}(x) + t_{\neg 🍫}(x)$

Formulas

First-Order Logic with Counting (FOC)

Counting terms

- $t_{🍫}(x) = \#(c).(Brought(x, c) \wedge Type(c, 🍫))$
- $t_{\neg 🍫}(x) = \#(c).(Brought(x, c) \wedge \neg Type(c, 🍫))$
- $p(x) = 2 \cdot t_{🍫}(x) + t_{\neg 🍫}(x)$

Formulas

- $\varphi_1(x) = t_{🍫}(x) \leqslant t_{\neg 🍫}(x)$
- $\varphi_2(x) = \forall y \, (t_{🍫}(y) \leqslant t_{🍫}(x))$

First-Order Logic with Counting (FOC)

Counting terms

- $t_{🍫}(x) = \#(c).(Brought(x,c) \wedge Type(c, 🍫))$
- $t_{\neg 🍫}(x) = \#(c).(Brought(x,c) \wedge \neg Type(c, 🍫))$
- $p(x) = 2 \cdot t_{🍫}(x) + t_{\neg 🍫}(x)$
- $q = \#(x).\varphi_1(x)$

Formulas

- $\varphi_1(x) = t_{🍫}(x) \leqslant t_{\neg 🍫}(x)$
- $\varphi_2(x) = \forall y\, (t_{🍫}(y) \leqslant t_{🍫}(x))$

First-Order Logic with Counting (FOC)

Counting terms

- $t_{\text{🍫}}(x) = \#(c).(Brought(x,c) \wedge Type(c, \text{🍫}))$
- $t_{\neg\text{🍫}}(x) = \#(c).(Brought(x,c) \wedge \neg Type(c, \text{🍫}))$
- $p(x) = 2 \cdot t_{\text{🍫}}(x) + t_{\neg\text{🍫}}(x)$
- $q = \#(x).\varphi_1(x)$

Formulas

- $\varphi_1(x) = t_{\text{🍫}}(x) \leqslant t_{\neg\text{🍫}}(x)$
- $\varphi_2(x) = \forall y \, (t_{\text{🍫}}(y) \leqslant t_{\text{🍫}}(x))$

FOC_1

- introduced by Grohe and Schweikardt (PODS 2018)
- subformulas comparing terms have at most one free variable
- has Gaifman-style normal forms

Main result

v. B. and Schweikardt, ICDT 2025

For integer-valued concepts definable in the first-order logic with counting FOC_1, there is an algorithm for the learning problem with

- **precomputation phase** in $n \cdot d^{\mathcal{O}(1)}$
- **learning phase** in $(d + t)^{\mathcal{O}(1)}$

(n: size of active domain, d: degree of the database, t: number of examples)

Main result

v. B. and Schweikardt, ICDT 2025

For integer-valued concepts definable in the first-order logic with counting FOC_1, there is an algorithm for the learning problem with

- **precomputation phase** in $n \cdot d^{\mathcal{O}(1)}$
- **learning phase** in $(d + t)^{\mathcal{O}(1)}$

(n: size of active domain, d: degree of the database, t: number of examples)

For databases of polylogarithmic degree:

- **precomputation phase** in quasi-linear time $n \cdot (\log n)^{\mathcal{O}(1)}$
- **learning phase** in time polylogarithmic in the size of the database $(\log n + t)^{\mathcal{O}(1)}$

Discussion

v. B. and Schweikardt, ICDT 2025

Integer-valued concepts definable in first-order logic with counting can be learned in polylog time after quasi-linear-time precomputation.

Discussion

v. B. and Schweikardt, ICDT 2025

Integer-valued concepts definable in first-order logic with counting can be learned in polylog time after quasi-linear-time precomputation.

- data complexity vs. parameterised complexity
- consistent learning vs. probably approximately correct (PAC) learning
- logics with weight aggregation

Bring more (chocolate) cakes!