Updated 2/4/2022

Topic 1: Data models and query languages
Unit 2: Logic & relational calculus (continued)
Lecture 05

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp22)

https://northeastern-datalab.github.io/cs7240/sp22/

2/1/2022

Pre-class conversations

- Last class recapitulation
- Project ideas
- Slides from today likely only posted tomorrow night. Sorry!

- today:
  - logic continued (likely next time algebra and the connection)
  - logic becomes super important today again, lack of expertise; thus lots of practice today ☺
  - in particular the concept of "undecidability": intuition for why things can quickly get complicated without giving proofs

Topic 1: Data Models and Query Languages

PRELIMINARY

- **Lecture 1 (Tue 1/18):** Course introduction, SQL, PostgreSQL setup, SQL Activities
- **Lecture 2 (Fri 1/21):** SQL
- **Lecture 3 (Tue 1/25):** SQL
- **Lecture 4 (Fri 1/28):** Logic & Relational Calculus
- **Lecture 5 (Tue 2/1):** Relational Algebra & Codd's Theorem
- **Lecture 6 (Fri 2/4):** Relational Algebra & Codd's Theorem
- **Lecture 7 (Tue 2/8):** Datalog & Recursion,
- **Lecture 8 (Fri 2/11):** Datalog & Recursion,
- **Lecture 9 (Tue 2/15):** Alternative Data Models

Pointers to relevant concepts & supplementary material:

- **Unit 1. SQL**: [SAMS'12], [CS 3200], [Cow'03] Ch3 & Ch5, [Complete'08] Ch6, [Silberschatz+'20] Ch3.8
- **Unit 2. Logic & Relational Calculus**: First-Order Logic (FOL), relational calculus (RC): [Barland+'08] 4.1.2 & 4.2.1 & 4.4, [Genesereth+] Ch6, [Halpern+'01], [Cow'03] Ch4.3 & 4.4, [Elmasri, Navathe'15] Ch8.6 & Ch8.7, [Silberschatz+'20] Ch27.1 & Ch27.2, [Alice'95] Ch3.1-3.3 & Ch4.2 & Ch4.4 & Ch5.3-5.4, [Barker-Plummer+'11] Ch11
- **Unit 3. Relational Algebra & Codd's Theorem**: Relational Algebra (RA), Codd's theorem: [Cow'03] Ch4.2, [Complete'08] Ch2.4 & Ch5.1-5.2, [Elmasri, Navathe'15] Ch8, [Silberschatz+'20] Ch2.6, [Alice'95] Ch4.4 & Ch5.4
- **Unit 4. Datalog & Recursion**: Datalog, recursion, Stratified Datalog with negation, Datalog evaluation strategies, Stable Model semantics, Answer Set Programming (ASP): [Complete'08] Ch5.3, [Cow'03] Ch 24, [Koutris'19] L9 & L10, [G., Suciu'10]
- **Unit 5. Alternative Data Models**: NoSQL: [Hellerstein, Stonebraker'05], [Sadalage, Fowler'12], [Harrison'16]

Queries and the connection to logic

- Why logic?
- A crash course in FOL
- Relational Calculus
  - Syntax and Semantics
  - Domain Independence and Safety

Entire Story in One Slide

1. RC = FOL over DB
2. RC can express “bad queries” that depend not only on the DB, but also on the domain from which values are taken (domain dependence)
3. We cannot test whether an RC query is “good,” but we can use a ”good” subset of RC that captures all “good” queries (safety)
4. “Good” RC and RA can express the same queries! (equivalence = Codd's theorem)

Relational Calculus (RC)

- RC is, essentially, first-order logic (FOL) over the schema relations
  - A query has the form "find all tuples $(x_1,\ldots,x_k)$ that satisfy an FOL condition"
- RC is a declarative query language
  - Meaning: a query is not defined by a sequence of operations, but rather by a condition that the result should satisfy

Queries and the connection to logic

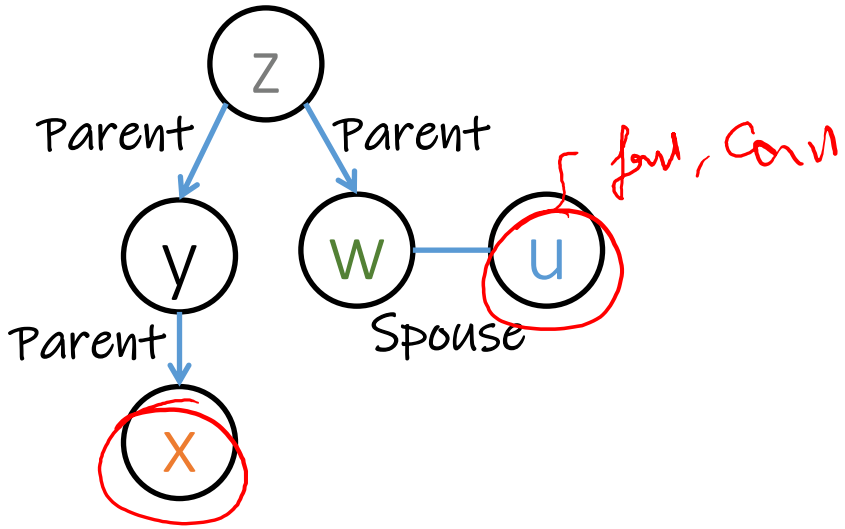
- Why logic?
- A crash course in FOL
- Relational Calculus
  - Syntax and Semantics
  - Domain Independence and Safety

RC Query

Person(id, gender, country)
Parent(parent, child)
Spouse(person1, person2)

assume symmetric relation
(a,b)∈Spouse ⇔ (b,a)∈Spouse

$$\{ (x,u) \mid \text{Person}(u, \text{'female'}, \text{'Canada'}) \wedge$$
$$\exists z,y \left[\text{Parent}(z,y) \wedge \text{Parent}(y,x) \wedge\right.$$
$$\left.\exists w \left[\text{Parent}(z,w) \wedge y \neq w \wedge (u=w \vee \text{Spouse}(u,w))\right]\right] \}$$



Which relatives does this query find?

?

RC Symbols

- **Constant values:** a, b, c, female, Canada, ...
  - Values that may appear in table cells (optionally with quotation marks)
- **Variables:** x, y, z, ...
  - Range over the values that may appear in table cells
- **Relation symbols:** R, S, T, Person, Parent, ...
  - Each with a specified arity
  - Will be fixed by the relational schema at hand
  - No attribute names, only attribute positions (= unnamed perspective)!
- Unlike general FOL, no function symbols!

RC Formulas (atomic and non-atomic)

- Atomic formulas:
  - $R(t_1,...,t_k)$ — Person(x, 'female', 'Canada')
    - R is a k-ary relation, Each t_i is a variable or a constant
    - Semantically it states that $(t_1,...,t_k)$ is a tuple in R
  - x op u — x=y, y≠w, z>5, z='female'
    - x is a variable, u is a variable/constant, op is one of >, <, =, ≠
    - Simply binary predicates, predefined interpretation
- Formula:
  - Atomic formula
  - If ϕ and ψ are formulas then these are formulas:
    $\phi \wedge \psi \quad \phi \vee \psi \quad \phi \rightarrow \psi \quad \phi \rightarrow \psi \quad \neg\phi \quad \exists x\, \phi \quad \forall x\, \phi$

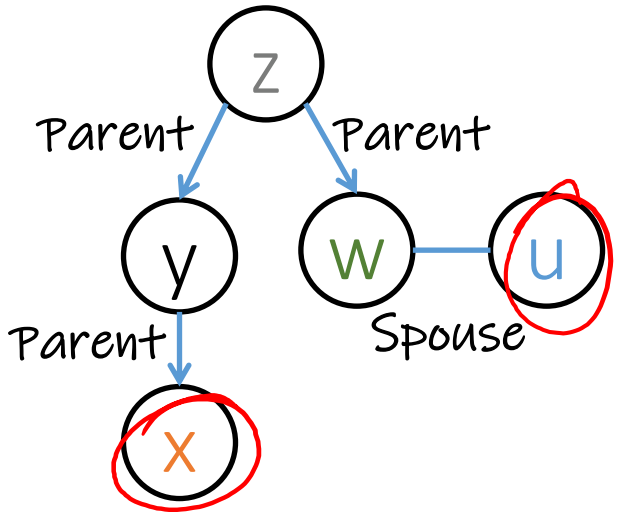
Free Variables

- Intuitively: free variable are not bound to quantifiers
- Formally:
  - A free variable of an atomic formula is a variable that occurs in the atomic formula
  - A free variable of $\varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi$ is a free variable of either φ or ψ
  - A free variable of $\neg\varphi$ is a free variable of φ
  - A free variable of $\exists x\ \varphi$ and $\forall x\ \varphi$ is a free variable y of φ such that $y \neq x$
- We write $\varphi(x_1,\ldots,x_k)$ to state that $x_1,\ldots,x_k$ are the free variables of formula φ (in some order)

Back to our earlier example

This is a formula!

$$\text{Person}(u, \text{'female'}, \text{'Canada'}) \wedge$$
$$\exists z,y \left[\text{Parent}(z,y) \wedge \text{Parent}(y,x) \wedge\right.$$
$$\left.\exists w \left[\text{Parent}(z,w) \wedge y \neq w \wedge (u = w \vee \text{Spouse}(u,w))\right]\right]$$

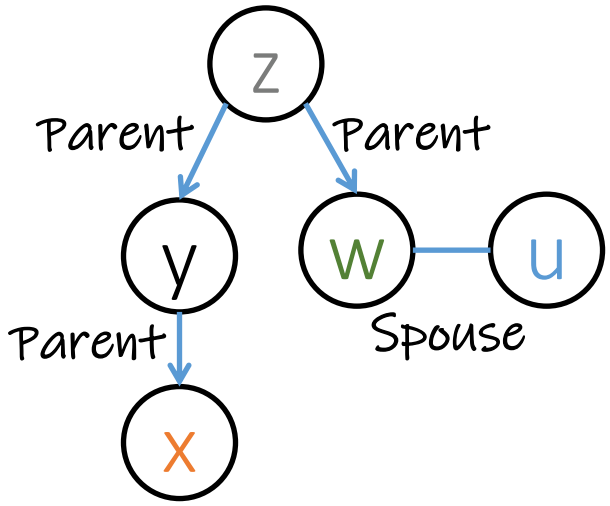


What are the free variables?

?

Back to our earlier example

$$\text{Person}(u, \text{'female'}, \text{'Canada'}) \wedge$$
$$\exists z,y \left[\text{Parent}(z,y) \wedge \text{Parent}(y,x) \wedge \right.$$
$$\left. \exists w \left[\text{Parent}(z,w) \wedge y \neq w \wedge (u = w \vee \text{Spouse}(u,w))\right]\right]$$

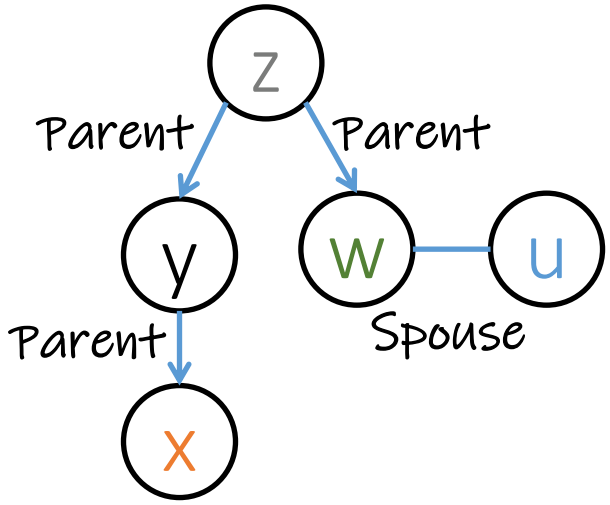


Notation:

$\varphi(x,u)$ / CanadianAunt(u,x)

RC query

$$\{ (x,u) \mid \text{Person}(u, \text{'female'}, \text{'Canada'}) \wedge$$
$$\exists z,y \, [\text{Parent}(z,y) \wedge \text{Parent}(y,x) \wedge$$
$$\exists w \, [\text{Parent}(z,w) \wedge y \neq w \wedge (u = w \vee \text{Spouse}(u,w))]] \}$$



$$\{ (x_1,\ldots,x_k) \mid \varphi(x_1,\ldots,x_k) \}$$

$$\varphi(x,u) \,/\, \text{CanadianAunt}(u,x)$$

Relation Calculus Query

some condition on the variables
COND(x_1,…,x_k)

- An RC query is an expression of the form

$$\{ (x_1,\ldots,x_k) \mid \varphi(x_1,\ldots,x_k) \}$$

where $\varphi(x_1,\ldots,x_k)$ is an RC formula

- An RC query is *over* a relational schema $\mathbf{S}$ if all the relation symbols belong to $\mathbf{S}$ (with matching arities)

DRC vs. TRC

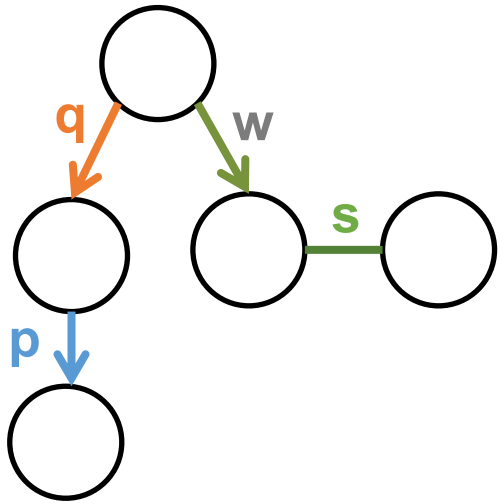
- There are two common variants of RC:
  - DRC: *Domain Relational Calculus* (what we have seen so far)
  - TRC: *Tuple Relational Calculus*
- DRC applies vanilla FO: terms interpreted as attribute values, relations have arity but no attribute names (= unnamed perspective)
- TRC is more "database friendly": terms interpreted as tuples with named attributes
- There are easy conversions between the two formalisms

Our Example in TRC

Person(id, gender, country)
Parent(parent, child)
Spouse(person1, person2)

optionally "t(nephew, aunt)"

$$\{ t \mid \exists a \in \text{Person}\ [a.\text{gender} = \text{'female'} \wedge a.\text{country} = \text{'Canada'}] \wedge$$
$$\exists p,q,w \in \text{Parent}\ [p.\text{child} = t.\text{nephew} \wedge q.\text{child} = p.\text{parent} \wedge$$
$$w.\text{parent} = q.\text{parent} \wedge w.\text{child} \neq q.\text{child} \wedge a.\text{id} = t.\text{aunt} \wedge$$
$$(w.\text{child} = a.\text{id} \vee \exists s\ [s \in \text{Spouse} \wedge s.\text{person1} = w.\text{child} \wedge s.\text{person2} = a.\text{id}])\ ]\ ]\}$$



tuple variables like in SQL instead of domain variables: {t | COND(t)}

often used short forms:

$\forall x \in R[\varphi]$ same as $\forall x[x \in R \Rightarrow \varphi]$

$\exists x \in R[\varphi]$ same as $\exists x[x \in R \wedge \varphi]$

Based on material by Benny Kimelfeld and Oded Shmueli for 236363 Database Management Systems, Technion, 2018. However, notice I prefer and follow here the notation of [Ramakrishnan, Gehrke' 03] and [Elmasri, Navathe'15] of using **a.country = 'Canada'**, instead of the alternative notation **a[country]='Canada'** used by [Silberschatz, Korth, Sudarshan 2010]

Different TRC notations

Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink)

331

Find persons that frequent only bars that serve only drinks they like.
(Find persons who like all drinks that are served in all the bars they visit.)
(Find persons for which there does not exist a bar they frequent that serves a drink they do not like.)

{q(person) | ∃f ∈ Frequents [f.person=q.person ∧ ¬(∃f2 ∈ Frequents [f2.person=f.person ∧ ¬(∃l ∈ Likes, ∃s ∈ Serves [l.drink=s.drink ∧ f2.bar=s.bar ∧ f2.person=l.person])])]}

my preferred notation

{F.person | F ∈ Frequents.(∄F2 ∈ Frequents.(F2.person=F.person ∧ (∄L ∈ Likes, ∄S ∈ Serves.(L.drink=S.drink ∧ F2.bar=S.bar ∧ F2.person=L.person))))}

my earlier preferred notation

{t: Person | ∃f ∈ Frequents [t(Person)=f(Person) ∧ ¬∃f2 ∈ Frequents [F2(person)=F(person) ∧ ¬(∃l ∈ Likes ∃s ∈ Serves) [l(Drink)=s(Drink) ∧ f2(Bar)=s(Bar) ∧ f2(Person)=l(Person)]]]}

[Deutsch 2019]

{f.Person | Frequents(f) AND (NOT(∃f2)(Frequents(f2) AND f2.person=f.person ∧ (NOT(∃l)(∃s)(Likes(l) AND Serves(s) AND l.drink=s.drink AND f2.bar=s.bar AND f2.person=l.person))))}

[Elmasri 2015]

$\{\mu^{(1)} \mid (\exists\rho^{(2)})\ (\text{Frequents}(\rho) \wedge \rho[1]=\mu[1] \wedge \neg((\exists\lambda^{(2)})(\text{Frequents}(\lambda) \wedge \lambda[1]=\rho[1] \wedge \neg((\exists\nu^{(2)})(\exists\theta^{(2)})(\text{Likes}(\nu) \wedge \text{Serves}(\theta) \wedge \nu(2)=\theta(2) \wedge \lambda(2)=\theta(1) \wedge \lambda(1)=\nu(1)))))\}$

[Ullman 1988]

{P| ∃F ∈ Frequents (F.person=P.person ∧ ¬∃F2 ∈ Frequents(F2.person=F.person ∧ ¬(∃L ∈ Likes ∃S ∈ Serves (L.drink=S.drink ∧ F2.bar=S.bar ∧ F2.person=L.person))))}

[Ramakrishnan 2003]

Queries and the connection to logic

- Why logic?
- A crash course in FOL
- Relational Calculus
  - Syntax and Semantics
  - Domain Independence and Safety

Bringing in the Domain

- Let $\mathbf{S}$ be a schema, D a database over $\mathbf{S}$, and Q an RC query over $\mathbf{S}$
- Then D gives an interpretation for the underlying FOL
  - Predicates ⟶ relations; constants copied; no functions

Is this true ?

Bringing in the Domain

- Let $\mathbf{S}$ be a schema, D a database over $\mathbf{S}$, and Q an RC query over $\mathbf{S}$
- ~~Then D gives an interpretation for the underlying FOL~~
  - Predicates ⟶ relations; constants copied; no functions
  - Not yet! We need to answer first: What is the domain?

- The active domain $\mathbf{ADom}$ (of D and Q) is the set of all the values that occur in either D or Q
- The query Q is evaluated over D with respect to a domain $\mathbf{Dom}$ that contains the active domain ($\mathbf{Dom} \supseteq \mathbf{ADom}$)
- Denote by $\mathrm{Q}^{\mathbf{Dom}}(\mathrm{D})$ the result of evaluating Q over D relative to the domain $\mathbf{Dom}$

Domain Independence

- Let $\mathbf{S}$ be a schema, and let Q be an RC query over $\mathbf{S}$
- We say that Q is domain independent if for every database D over $\mathbf{S}$ and ...

How could we continue the definition ?

Domain Independence

- Let $\mathbf{S}$ be a schema, and let Q be an RC query over $\mathbf{S}$
- We say that Q is domain independent if for every database D over $\mathbf{S}$ and every two domains $\mathbf{Dom_1}$ and $\mathbf{Dom_2}$ that contain the active domain, we have:

$$Q^{\mathbf{Dom_1}}(D) = Q^{\mathbf{Dom_2}}(D) = Q^{\mathbf{ADom}}(D)$$

Bad News...

- We would like be able to tell whether a given RC query is domain independent, and then reject “bad queries”
- Alas, this problem is undecidable!
  - That is, there is no algorithm that takes as input an RC query and returns true iff the query is domain independent

Based on material by Benny Kimelfeld and Oded Shmueli for 236363 Database Management Systems, Technion, 2018.
First observed in "Di Paola. The Recursive Unsolvability of the Decision Problem for the Class of Definite Formulas, JACM 1969. https://doi.org/10.1145/321510.321524"
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Good News

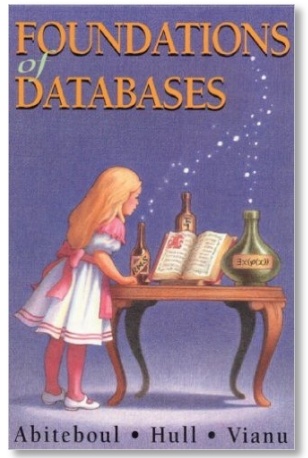
Domain-independent RC has an "effective syntax", that is:

- A syntactic restriction of RC in which every query is domain independent
- Restricted queries are said to be safe

- Safety can be tested automatically (and efficiently)
  - Most importantly, for every domain independent RC query there exists an equivalent safe RC query!

Safety

- We do not formally define the safe syntax in this course
- Details on the safe syntax can be found in Ch 5.4 of [Alice'95]: Foundations of Databases by Abiteboul, Hull and Vianu
  - Example:
    - In $\exists x\, \varphi$, the variable x should be guarded by φ
    - Every variable x_i is guarded by $R(x_1,\ldots,x_k)$
    - In $\varphi \wedge (x=y)$, the variable x is guarded if and only if either x or y is guarded by φ
    - ... and so on



Based on material by Benny Kimelfeld and Oded Shmueli for 236363 Database Management Systems, Technion, 2018.
An accessible overview of issues involving safety is: Topor, Safety and Domain Independence, Encyclopedia of Database Systems. https://doi.org/10.1007/978-0-387-39940-9_1255
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

ADom = {1, 2, 3, 'female', 'Canada'}

Dom = ADom ∪ {'elefant', 'car', 'lemon', π, ...}

$$\{ (x) \mid \neg \text{Person}(x, \text{'female'}, \text{'Canada'}) \}$$?

$$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y = z] \}$$?

$$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y \neq z] \}$$?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

Example fixes: ... ∧ ∃y,z.Person(x,y,z)
... ∧ ∃y.Person(x,y,'Canada')

$\{ (x) \mid \neg \text{Person}(x, \text{'female'}, \text{'Canada'}) \}$

x could be also 'Canada' or 'female' or ...

Not DI

$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y = z] \}$?

$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y \neq z] \}$?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

Example fixes: ... ∧ ∃y,z.Person(x,y,z)
... ∧ ∃y.Person(x,y,'Canada')

$$\{ (x) \mid \neg \text{Person}(x, \text{'female'}, \text{'Canada'}) \}$$

Not DI

x could be also 'Canada' or 'female' or ...

$$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y = z] \}$$

DI

same as {(x,y) | Spouse(x,y)} = Spouse(x,y)

$$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y \neq z] \}$$

?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

Example fixes: ... ∧ ∃y,z.Person(x,y,z)
... ∧ ∃y.Person(x,y,'Canada')

$$\{ (x) \mid \neg \text{Person}(x, \text{'female'}, \text{'Canada'}) \}$$

x could be also 'Canada' or 'female' or ...

Not DI

$$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y=z] \}$$

same as {(x,y) | Spouse(x,y)}

DI

$$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y\neq z] \}$$

?

D: Spouse('Alice','Bob')
ADom={'Alice','Bob'} → {('Alice','Alice')}
Dom ⊇ ADom Dom={'Alice','Bob','Charly'} → {('Alice','Alice'), ('Alice','Charly')}

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

Example fixes: ... ∧ ∃y,z.Person(x,y,z)
... ∧ ∃y.Person(x,y,'Canada')

$$\{ (x) \mid \neg \text{Person}(x, \text{'female'}, \text{'Canada'}) \}$$

Not DI

x could be also 'Canada' or 'female' or ...

$$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y=z] \}$$

DI

same as {(x,y) | Spouse(x,y)}

$$\{ (x,y) \mid \exists z\, [\text{Spouse}(x,z) \wedge y\neq z] \}$$

Not DI

D: Spouse('Alice','Bob')
ADom={'Alice','Bob'} → {('Alice','Alice')}
Dom ⊇ ADom Dom={'Alice','Bob','Charly'} → {('Alice','Alice'), ('Alice','Charly')}

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

$$\{ (x) \mid \exists z,w \; \text{Person}(x,z,w) \wedge \exists y \; [\neg \text{Likes}(x,y)] \}$$

$$\{ (x) \mid \exists z,w \; \text{Person}(x,z,w) \wedge \forall y \; [\neg \text{Likes}(x,y)] \}$$

$$\{ (x) \mid \exists z,w \; \text{Person}(x,z,w) \wedge \forall y \; [\neg \text{Likes}(x,y)] \wedge \exists y \; [\neg \text{Likes}(x,y)] \}$$

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'female', 'Canada')
Person('Beate', 'female', 'Canada')
Person('Cecile', 'female', 'Canada')

Likes('Alice', 'Beate')
Likes('Alice', 'Cecile')
Likes('Alice', 'Alice')

ADom = ?

$$\{ (x) \mid \exists z,w \ \text{Person}(x,z,w) \wedge \exists y \ [\neg \text{Likes}(x,y)] \}$$

$$\{ (x) \mid \exists z,w \ \text{Person}(x,z,w) \wedge \forall y \ [\neg \text{Likes}(x,y)] \}$$

$$\{ (x) \mid \exists z,w \ \text{Person}(x,z,w) \wedge \forall y \ [\neg \text{Likes}(x,y)] \wedge \exists y \ [\neg \text{Likes}(x,y)] \}$$

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'female', 'Canada')
Person('Beate', 'female', 'Canada')
Person('Cecile', 'female', 'Canada')

Likes('Alice', 'Beate')
Likes('Alice', 'Cecile')
Likes('Alice', 'Alice')

ADom = {'Alice', 'Beate', 'Cecile', 'female', 'Canada')

$$\{ (x) \mid \exists z,w \ \text{Person}(x,z,w) \wedge \exists y \ [\neg \text{Likes}(x,y)] \}$$

$$\{ (x) \mid \exists z,w \ \text{Person}(x,z,w) \wedge \forall y \ [\neg \text{Likes}(x,y)] \}$$

$$\{ (x) \mid \exists z,w \ \text{Person}(x,z,w) \wedge \forall y \ [\neg \text{Likes}(x,y)] \wedge \exists y \ [\neg \text{Likes}(x,y)] \}$$

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'Alice', 'Alice')
Person('Beate', 'Beate', 'Beate')
Person('Cecile', 'Beate', 'Beate')

Likes('Alice', 'Beate')
Likes('Alice', 'Cecile')
Likes('Alice', 'Alice')

ADom = {'Alice', 'Beate', 'Cecile')

Dom = {'Alice', 'Beate', 'Cecile', 'Dora')

$$\{ (x) \mid \exists z,w \ \text{Person}(x,z,w) \wedge \exists y \ [\neg \text{Likes}(x,y)] \}$$?

$$\{ (x) \mid \exists z,w \ \text{Person}(x,z,w) \wedge \forall y \ [\neg \text{Likes}(x,y)] \}$$?

$$\{ (x) \mid \exists z,w \ \text{Person}(x,z,w) \wedge \forall y \ [\neg \text{Likes}(x,y)] \wedge \exists y \ [\neg \text{Likes}(x,y)] \}$$?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'Alice', 'Alice')	Likes('Alice', 'Beate')
Person('Beate', 'Beate', 'Beate')	Likes('Alice', 'Cecile')
Person('Cecile', 'Beate', 'Beate')	Likes('Alice', 'Alice')

ADom = {'Alice', 'Beate', 'Cecile')

Dom = {'Alice', 'Beate', 'Cecile', 'Dora')

Example fix: ?

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \exists y\ [\neg \text{Likes}(x,y)] \}$$

Not DI

Alice is in the output if Dom ⊃ ADom (e.g., Dora is in Dom)

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \forall y\ [\neg \text{Likes}(x,y)] \}$$?

$$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \forall y\ [\neg \text{Likes}(x,y)] \wedge \exists y\ [\neg \text{Likes}(x,y)] \}$$?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'Alice', 'Alice')
Person('Beate', 'Beate', 'Beate')
Person('Cecile', 'Beate', 'Beate')

Likes('Alice', 'Beate')
Likes('Alice', 'Cecile')
Likes('Alice', 'Alice')

ADom = {'Alice', 'Beate', 'Cecile')

Dom = {'Alice', 'Beate', 'Cecile', 'Dora')

Person(y,_,_)

Example fix: ... ∧ ∃u,v [Person(y,u,v)]

$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \exists y\ [\neg \text{Likes}(x,y)] \}$

Not DI

Alice is in the output if Dom ⊃ ADom (e.g., Dora is in Dom)

$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \forall y\ [\neg \text{Likes}(x,y)] \}$?

$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \forall y\ [\neg \text{Likes}(x,y)] \wedge \exists y\ [\neg \text{Likes}(x,y)] \}$?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'Alice', 'Alice')	Likes('Alice', 'Beate')
Person('Beate', 'Beate', 'Beate')	Likes('Alice', 'Cecile')
Person('Cecile', 'Beate', 'Beate')	Likes('Alice', 'Alice')

ADom = {'Alice', 'Beate', 'Cecile')

Dom = {'Alice', 'Beate', 'Cecile', 'Dora')

Person(y,_,_)

Example fix: ... ∧ ∃u,v [Person(y,u,v)]

$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \exists y\ [\neg \text{Likes}(x,y)] \}$

Not DI

Alice is in the output if Dom ⊃ ADom (e.g., Dora is in Dom)

$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \forall y\ [\neg \text{Likes}(x,y)] \}$

DI

x never occurs in Likes(x,_): Beate, Cecile

$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \forall y\ [\neg \text{Likes}(x,y)] \wedge \exists y\ [\neg \text{Likes}(x,y)] \}$

?

Which One is Domain Independent?

Person(id, gender, country)
Likes(person1, person2)
Spouse(person1, person2)

D

Person('Alice', 'Alice', 'Alice')	Likes('Alice', 'Beate')
Person('Beate', 'Beate', 'Beate')	Likes('Alice', 'Cecile')
Person('Cecile', 'Beate', 'Beate')	Likes('Alice', 'Alice')

ADom = {'Alice', 'Beate', 'Cecile')

Dom = {'Alice', 'Beate', 'Cecile', 'Dora')

Person(y,_,_)

Example fix: ... ∧ ∃u,v [Person(y,u,v)]

$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \exists y\ [\neg \text{Likes}(x,y)] \}$ — Not DI

Alice is in the output if Dom ⊃ ADom (e.g., Dora is in Dom)

$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \forall y\ [\neg \text{Likes}(x,y)] \}$ — DI

x never occurs in Likes(x,_): Beate, Cecile

$\{ (x) \mid \exists z,w\ \text{Person}(x,z,w) \wedge \forall y\ [\neg \text{Likes}(x,y)] \wedge \exists y\ [\neg \text{Likes}(x,y)] \}$ — DI

implication (absorption) if Dom ≠ ∅, which is necessary for there to be Person(x,_,_)

What is the meaning of following unsafe expressions?

$\{ x \mid \exists y.\ R(x) \}$?

$\{ x \mid x \geq 10 \}$?

$\{ x \mid \forall y\ R(x,y) \}$?

What is the meaning of following unsafe expressions?

$\{ x \mid \exists y.\ R(x) \}$ logically equivalent to $\{ x \mid R(x) \} = R(x)$

$\{ x \mid x \geq 10 \}$?

$\{ x \mid \forall y\ R(x,y) \}$?

What is the meaning of following unsafe expressions?

$\{ x \mid \exists y.\ R(x) \}$ logically equivalent to $\{ x \mid R(x) \} = R(x)$

$\{ x \mid x \geq 10 \}$ What if Dom=$\mathbb{N}$? DI: $\{ x \mid A(x) \wedge x \geq 10 \}$

$\{ x \mid \forall y\ R(x,y) \}$?

What is the meaning of following unsafe expressions?

$\{ x \mid \exists y.\ R(x) \}$ logically equivalent to $\{ x \mid R(x) \} = R(x)$

$\{ x \mid x \geq 10 \}$ What if Dom=ℕ? DI: $\{ x \mid A(x) \wedge x \geq 10 \}$

$\{ x \mid \forall y\ R(x,y) \}$ D: R('a','a')
ADom={'a'}
Dom={'a','Chile'}

DI ?: $\{ x \mid \forall y\ [A(y) \rightarrow R(x,y)] \}$

 ?

What is the meaning of following unsafe expressions?

$\{ x \mid \exists y.\ R(x) \}$ logically equivalent to $\{ x \mid R(x) \} = R(x)$

$\{ x \mid x \geq 10 \}$ What if Dom=ℕ?

DI: $\{ x \mid A(x) \wedge x \geq 10 \}$

$\{ x \mid \forall y\ R(x,y) \}$

D: R('a','a')
ADom={'a'}
Dom={'a','Chile'}

DI ?: $\{ x \mid \forall y\ [A(y) \rightarrow R(x,y)] \}$

what if relation A is empty?

?

What is the meaning of following unsafe expressions?

$\{ x \mid \exists y.\ R(x) \}$ logically equivalent to $\{ x \mid R(x) \} = R(x)$

$\{ x \mid x \geq 10 \}$ What if Dom=ℕ?

DI: $\{ x \mid A(x) \wedge x \geq 10 \}$

$\{ x \mid \forall y\ R(x,y) \}$

D: R('a','a')
ADom={'a'}
Dom={'a','Chile'}

DI ?: $\{ x \mid \forall y\ [A(y) \rightarrow R(x,y)] \}$

what if relation A is empty?

1. always true for A=∅

$\{ x \mid \forall y\ [\neg A(y) \vee R(x,y)] \}$

What is the meaning of following unsafe expressions?

$\{ x \mid \exists y.\, R(x) \}$ logically equivalent to $\{ x \mid R(x) \} = R(x)$

$\{ x \mid x \geq 10 \}$ What if Dom=ℕ? DI: $\{ x \mid A(x) \wedge x \geq 10 \}$

$\{ x \mid \forall y\, R(x,y) \}$ D: R('a','a') ADom={'a'} Dom={'a','Chile'}

not DI: $\{ x \mid \forall y\, [A(y) \rightarrow R(x,y)] \}$

what if relation A is empty?

Neutral element for ∀ is TRUE

$\sum$: $0 + x = x$

$\prod$: $1 \cdot x = x$

$\vee$: $\text{FALSE} \vee x = x$ $\quad \exists : x_1 \vee x_2 \vee \ldots \vee \text{FALSE}$

$\wedge$: $\text{TRUE} \wedge x = x$ $\quad \forall : x_1 \wedge x_2 \wedge \ldots \wedge \text{TRUE}$

MIN: $\text{MIN}(\infty, x) = x$

1. always true for A=∅

$\{ x \mid \forall y\, [\neg A(y) \vee R(x,y)] \}$

2. alternative way to see that

What is the meaning of following unsafe expressions?

$\{ x \mid \exists y.\ R(x) \}$ logically equivalent to $\{ x \mid R(x) \} = R(x)$

$\{ x \mid x \geq 10 \}$ What if Dom=ℕ? DI: $\{ x \mid A(x) \wedge x \geq 10 \}$

$\{ x \mid \forall y\ R(x,y) \}$ D: R('a','a') ADom={'a'} Dom={'a','Chile'}

not DI: $\{ x \mid \forall y\ [A(y) \rightarrow R(x,y)] \}$

what if relation A is empty?

Neutral element for ∀ is **TRUE**

another way to see it:

$\forall y\ [R(y)]$

true if the domain for y is empty set!

$\forall y\ [y \in Dom \rightarrow R(y)]$

1. always true for A=∅

$\{ x \mid \forall y\ [\neg A(y) \vee R(x,y)] \}$

2. alternative way to see that

What is the meaning of following unsafe expressions?

$\{ x \mid \exists y.\, R(x) \}$ logically equivalent to $\{ x \mid R(x) \} = R(x)$

$\{ x \mid x \geq 10 \}$ What if Dom=$\mathbb{N}$? DI: $\{ x \mid A(x) \wedge x \geq 10 \}$

$\{ x \mid \forall y\, R(x,y) \}$ not DI: $\{ x \mid \forall y\, [A(y) \rightarrow R(x,y)] \}$

DI: ?

What is the meaning of following unsafe expressions?

$\{ x \mid \exists y.\ R(x) \}$ logically equivalent to $\{ x \mid R(x) \} = R(x)$

$\{ x \mid x \geq 10 \}$ What if Dom=$\mathbb{N}$? DI: $\{ x \mid A(x) \wedge x \geq 10 \}$

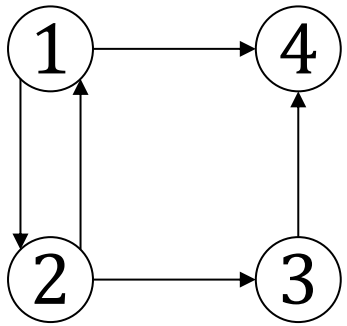
$\{ x \mid \forall y\ R(x,y) \}$ not DI: $\{ x \mid \forall y\ [A(y) \rightarrow R(x,y)] \}$

or $R(x,\_)$ $\exists z[R(x,z) \wedge \ldots]$

DI: $\{ x \mid A(x) \wedge \forall y\ [A(y) \rightarrow R(x,y)] \}$

$\{ x \mid A(x) \wedge \nexists y\ [A(y) \wedge \neg R(x,y)] \}$

Example: Querying a Graph



A:

1	2
2	1
2	3
1	4
3	4

A encodes the directed edges of a graph ("arcs")

What do these queries return ?

$$\{ x \mid \exists y.\, A(x,y) \}$$

?

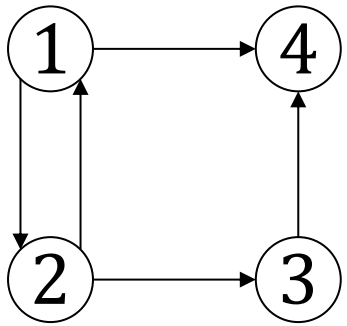
$$\{ x \mid \exists y,z,u.[A(x,y) \wedge A(y,z) \wedge A(z,u)] \}$$

?

$$\{ (x,y) \mid \forall z.[A(x,z) \rightarrow A(y,z)] \}$$

?

Example: Querying a Graph



A:

1	2
2	1
2	3
1	4
3	4

A encodes the directed edges of a graph ("arcs")

What do these queries return ?

$$\{ x \mid \exists y.\ A(x,y) \}$$

Nodes that have at least one child: ?

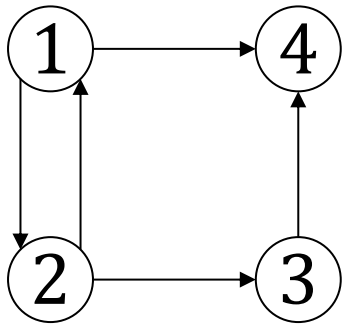
$$\{ x \mid \exists y,z,u.[A(x,y) \wedge A(y,z) \wedge A(z,u)] \}$$

?

$$\{ (x,y) \mid \forall z.[A(x,z) \rightarrow A(y,z)] \}$$

?

Example: Querying a Graph



A:

1	2
2	1
2	3
1	4
3	4

A encodes the directed edges of a graph ("arcs")

What do these queries return ?

$$\{ x \mid \exists y.\, A(x,y) \}$$

Nodes that have at least one child: {1,2,3}

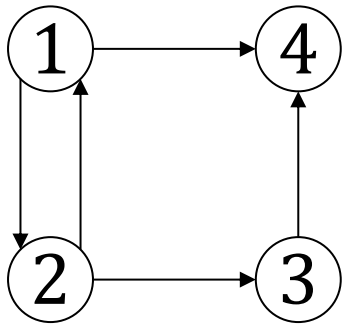
$$\{ x \mid \exists y,z,u.[A(x,y) \wedge A(y,z) \wedge A(z,u)] \}$$

?

$$\{ (x,y) \mid \forall z.[A(x,z) \rightarrow A(y,z)] \}$$

?

Example: Querying a Graph



A:

1	2
2	1
2	3
1	4
3	4

A encodes the directed edges of a graph ("arcs")

What do these queries return ?

$$\{ x \mid \exists y.\ A(x,y) \}$$

Nodes that have at least one child: {1,2,3}

$$\{ x \mid \exists y,z,u.[A(x,y) \wedge A(y,z) \wedge A(z,u)] \}$$

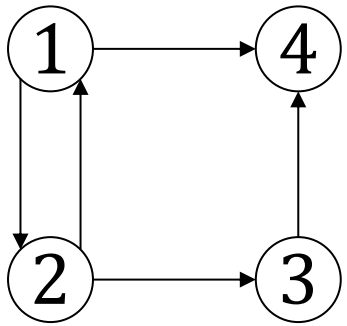
Nodes that have a great-grand-child: ?

$$\{ (x,y) \mid \forall z.[A(x,z) \rightarrow A(y,z)] \}$$

?

Based on an example by Dan Suciu from CSE 554, 2011.

Example: Querying a Graph



A:

1	2
2	1
2	3
1	4
3	4

A encodes the directed edges of a graph ("arcs")

What do these queries return ?

$$\{ x \mid \exists y.\ A(x,y) \}$$

Nodes that have at least one child: {1,2,3}

$$\{ x \mid \exists y,z,u.[A(x,y) \wedge A(y,z) \wedge A(z,u)] \}$$

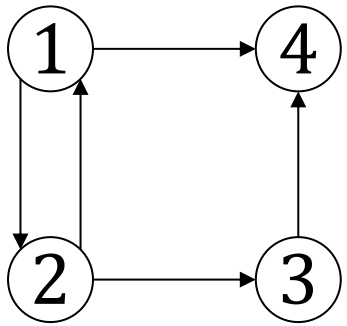
Nodes that have a great-grand-child: {1,2}

y≠u not necessary!
Contrast homomorphism vs. isomorphism ("Hamiltoninan Path")

$$\{ (x,y) \mid \forall z.[A(x,z) \rightarrow A(y,z)] \}$$

?

Example: Querying a Graph



A:

1	2
2	1
2	3
1	4
3	4

A encodes the directed edges of a graph ("arcs")

What do these queries return ?

$$\{ x \mid \exists y.\ A(x,y) \}$$

Nodes that have at least one child: {1,2,3}

$$\{ x \mid \exists y,z,u.[A(x,y) \wedge A(y,z) \wedge A(z,u)] \}$$

Nodes that have a great-grand-child: {1,2}

$\nexists z.[A(x,z) \wedge \neg A(y,z)]$

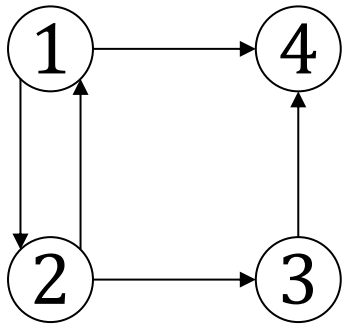
$$\{ (x,y) \mid \forall z.[A(x,z) \rightarrow A(y,z)] \}$$

Every child of x is a child of y.

(1,1) (4,4) (1,3) (3,1) (4,1)

Which of the following tuples fulfill the condition? ?

Example: Querying a Graph



A:		
	1	2
	2	1
	2	3
	1	4
	3	4

A encodes the directed edges of a graph ("arcs")

What do these queries return ?

$$\{ x \mid \exists y.\ A(x,y) \}$$

Nodes that have at least one child: {1,2,3}

$$\{ x \mid \exists y,z,u.[A(x,y) \wedge A(y,z) \wedge A(z,u)] \}$$

Nodes that have a great-grand-child: {1,2}

$$\{ (x,y) \mid \forall z.[A(x,z) \rightarrow A(y,z)] \}$$

$\nexists z.[A(x,z) \wedge \neg A(y,z)]$

Every child of x is a child of y.

Which of the following tuples fulfill the condition?

(1,1) (4,4) ~~(1,3)~~ (3,1) (4,1)

{(1,1),(2,2),(3,1),(3,3),(4,1), (4,2), (4,3), (4,4)}

The person/bar/drinks schema

Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink)

What does the following query compute?

$$\{ x \mid \forall y.[\text{Frequents}(x,y) \rightarrow \exists z.[\text{Serves}(y,z) \wedge \text{Likes}(x,z)]\}$$

?

Schema adapted from Jeff Ullman's drinkers/bars/beers example to avoid attributes with same first letters. https://dl.acm.org/doi/book/10.5555/42790
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

The person/bar/drinks schema

Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink)

What does the following query compute?

$$\{ x \mid \forall y.[\text{Frequents}(x,y) \rightarrow \exists z.[\text{Serves}(y,z) \wedge \text{Likes}(x,z)]\}$$

Find drinkers that frequent only bars
that serve some drink they like.

Is this query domain independent? ?

If not, how to fix?

Schema adapted from Jeff Ullman's drinkers/bars/beers example to avoid attributes with same first letters. https://dl.acm.org/doi/book/10.5555/42790

The person/bar/drinks schema

Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink)

What does the following query compute?

∃w.[Frequents(x,w) ∧ …
∃w.[Likes(x,w) ∧ …

Are those two options to make it safe identical ?

$$\{ x \mid \forall y.[\text{Frequents}(x,y) \rightarrow \exists z.[\text{Serves}(y,z) \wedge \text{Likes}(x,z)]] \}$$

Find drinkers that frequent only bars that serve some drink they like.

Careful! This query is not domain independent.

The person/bar/drinks schema

Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink)

What does the following query compute?

∃w.[Frequents(x,w) ∧ …
∃w.[Likes(x,w) ∧ …

Both safe, but not identical. Tip: Should a drinker who likes a drink but does not frequent a bar be returned?

$$\{ x \mid \forall y.[\text{Frequents}(x,y) \rightarrow \exists z.[\text{Serves}(y,z) \wedge \text{Likes}(x,z)]] \}$$

Find drinkers that frequent only bars that serve some drink they like.

Challenge: write this query without the ∀ quantifier!
And then in SQL

?

The person/bar/drinks example

Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink)

Challenge: write these in SQL.
Solutions at: https://demo.queryvis.com

Find persons that frequent some bar that serves some drink they like.

?

Find persons that frequent only bars that serve some drink they like.

$$\{ x \mid \forall y.[\text{Frequents}(x,y) \rightarrow \exists z.[\text{Serves}(y,z) \wedge \text{Likes}(x,z)]\}$$

Find persons that frequent some bar that serves only drinks they like.

?

Find persons that frequent only bars that serve only drinks they like.
(= Find persons who like all drinks that are served in all the bars they visit.)
(= Find persons for which there does not exist a bar they frequent that serves a drink they do not like.)

?