RELATIONAL CALCULUS

→ is a language for writing queries of the form $\{\bar{x} \mid \varphi\}$ where

the set of variables in $\bar{x}$ is **free(φ)** → All variables tht. are free in the body shld. be mentioned in the head!

Quantifiers are bound until the end of the line

i.e. $Q = \{x, y \mid \exists z (R(x,z) \wedge S(z,y))\}$

$Q = \{y, x \mid \exists z\, R(x,z) \wedge S(z,y)\}$

→ Same body, different head, hence they are different queries and will give you different answers!

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

↳ can have repetitions

→ Queries w/o free variables are called **boolean queries**

i.e. $Q = \{() \mid \forall x\, R(x,x)\}$ → Return an empty tuple whenever formula in body is satisfied — IF TRUE

↳ Return an empty set — IF FALSE

DATA MODEL (for RC)

→ **Relations** are sets of **tuples** of the same length → There is an order among the elems. of a row in a table

SCHEMA

- Consists of a set of **relation names**
- Each relation name has an **arity** (i.e. no. of columns)

INSTANCE

- Each relation name from the schema of arity k is associated w/ a k-ary relation

(i.e. a set of tuples that are all of length k)

EXAMPLES

Ternary Relations:
- Customer: ID, Name, Age
- Account: Num, Branch, CustID

1) Name of customers younger than 33 or older than 50

↳ $\{ y \mid \exists x, z \ \text{Customer}(x, y, z) \wedge (z < 33 \vee z > 50) \}$

(x: ID, y: Name, z: Age)

2) Name and age of customers having an account in London

↳ $\{ y, z \mid \exists x \ \text{Customer}(x, y, z) \wedge \exists w \ \text{Account}(w, \text{'London'}, x) \}$

Equivalent to:

$\{ y, z \mid \exists x, u, v \ \text{Customer}(x, y, z) \wedge \exists w \ \text{Account}(w, u, v) \wedge x = v \wedge u = \text{'London'} \}$

★ 3) ID of customers who have an account in every branch

↳ $\{ x \mid \exists y, z \ \text{Customer}(x, y, z) \wedge (\forall u, w, v \ \text{Account}(u, w, v) \rightarrow \exists u' \ \text{Account}(u', w, x)) \}$

- $\text{Account}(u, w, v)$: This part is used to go through the branches
- $\exists u'$: Use this to say that the acc. number is not necessarily the same

↳ Whenever there is an account in some branch w, there must also be an account in the same branch but is owned by the customer that I'm interested in.

INTERPRETATIONS

→ First Order structure $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ where $c^{\mathcal{I}} \in \Delta$ and $R^{\mathcal{I}} \subseteq \Delta^n$

→ In RC, every constant is interpreted as itself:

STANDARD NAME ASSUMPTION (SNA)

$$c^{\mathcal{I}} = c$$

↳ If we fix an underlying domain Δ under SNA, first-order structures are just databases

↳ bc. we removed the need to interpret constant symbols, hence the only thing we're interpreting are relations

ANSWERS TO QUERIES

→ Recall that an assignment ν maps variables to objects in Δ

→ The answer to a query $Q = \{\bar{x} \mid \varphi\}$ on a database D is

$\bar{x} = x_1, x_2, \ldots, x_n$

$$Q(D) = \{ \underbrace{\nu(\bar{x})}_{\nu(x_1), \nu(x_2), \ldots, \nu(x_n)} \mid \nu : \underbrace{\text{free}(\varphi)}_{\text{For all the variable assignments from the free variables of } \varphi \text{ to the domain } \Delta} \to \Delta \text{ such that } \underbrace{D, \nu \models \varphi}_{\text{first-order struct. } D \text{ and the variable assignment } \nu \text{ satisfies the formula } \varphi} \}$$

→ The answer to a boolean query is either $\{()\}$ (true) or $\emptyset$ (false)

SAFETY

→ A query is **safe** if it gives a finite answer on ALL databases and this answer does not depend on the universe Δ

Examples of UNSAFE queries:

- $\{x \mid \neg R(x)\}$
- $\{x, y \mid R(x) \vee R(y)\} = Q$
- $\{x, y \mid x = y\} = S$

i.e. $\Delta = \mathbb{N}^+$, $R^D = \{1\}$

$Q(D) = \{(1,1), (1,2), (1,3), \ldots\} \cup \{(1,1), (2,1), (3,1), \ldots\}$

(Satisfy $R(x)$ so value of y can be whatever; Satisfy $R(y)$ so value of x can be whatever)

$S(D) = \{(1,1), (2,2), (3,3), \ldots\}$

→ Boolean queries always give you finite answer ($\{()\}$ or $\emptyset$) BUT their answers may depend on Δ

i.e.

$\Delta = \{1,2,3\}$, $R^D = \{1,2,3\}$, $Q = \{() \mid \forall x\, R(x)\}$

$\therefore Q(D) = \{()\}$ (True)

if we change → $\Delta = \{1,2,3,4\}$ (4: not in the interpretation of R), $R^D = \{1,2,3\}$, $Q = \{() \mid \forall x\, R(x)\}$

$\therefore Q(D) = \{\}$ (False)

→ However, whether a RC query is safe is undecidable.

↳ SOLN: Active Domain

ACTIVE DOMAIN

$Adom(R)$ = { all constants occuring in R }

i.e.

$$Adom\left(\begin{array}{c|cc} R & A & B \\ \hline & a_1 & b_1 \\ & a_1 & b_2 \end{array}\right) = \{a_1, b_1, b_2\}$$

→ The active domain of a database D is

$$Adom(D) = \bigcup_{R \in D} Adom(R)$$ → Union of the active domain of relations in the database

$$= \{ \text{all constants occuring in } D \}$$

→ used to define the active domain semantics of RC

- The idea is that instead of evaluating queries within the universe, we evaluate them within the active domain of the database on which the query is asked → Safe Relational Calculus

$$Q(D) = \{ \nu(\bar{x}) \mid \nu : \mathbf{free}(\varphi) \to \mathbf{Adom}(D) \text{ s.t. } D, \nu \models \varphi \}$$

EVALUATION OF QUANTIFIERS

$$D, \nu \models \exists x\, \varphi \iff D, \nu \models \bigvee_{a \in Adom(D)} \varphi[x/a]$$

$$D, \nu \models \forall x\, \varphi \iff D, \nu \models \bigwedge_{a \in Adom(D)} \varphi[x/a]$$

disjunction / conjunction

$\varphi[x/a]$ denotes the formula obtained from φ by replacing all free occurences of x with a

i.e. Assume $Adom(D) = \{1, 2, 3\}$

- $D, \nu \models \exists x\, R(x,y) \wedge S(x)$

$$\iff$$

$$D, \nu \models (R(1,y) \wedge S(1)) \vee (R(2,y) \wedge S(2)) \vee (R(3,y) \wedge S(3))$$

- $D, \nu \models \forall x\, S(x) \to R(x,y)$

$$\iff$$

$$D, \nu \models (S(1) \to R(1,y)) \wedge (S(2) \to R(2,y)) \wedge (S(3) \to R(3,y))$$