Database Theory

VU 181.140, SS 2018

3. Codd's Theorem

Reinhard Pichler

Institute of Logic and Computation
DBAI Group
TU Wien

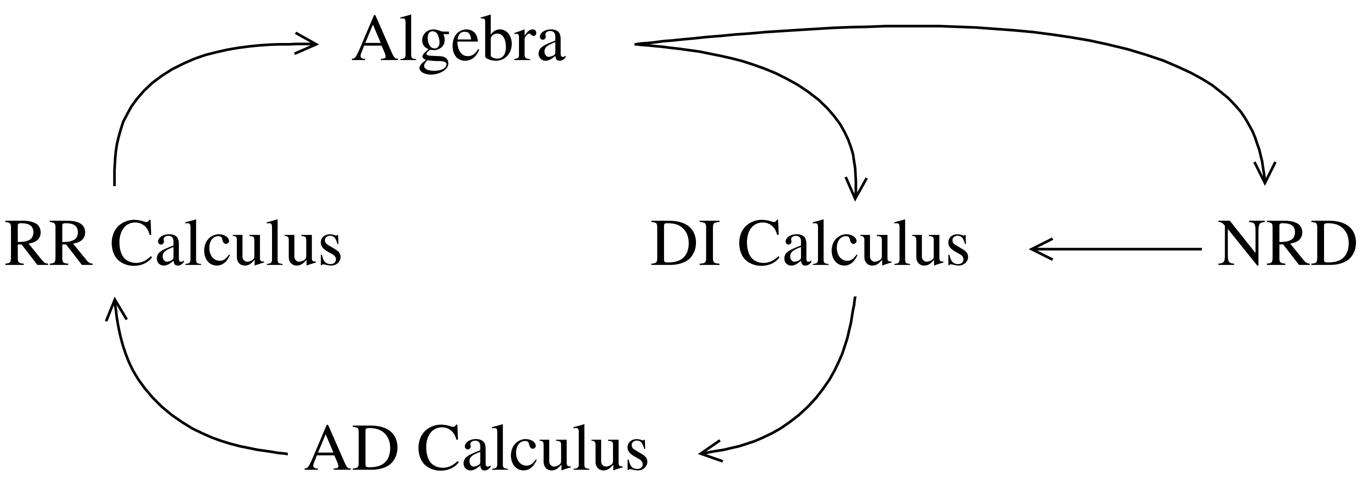
20 March, 2018

Outline

What is Codd's Theorem?

- Several query languages have been considered for relational databases:
  - some are more convenient for query specification
  - some are easier to optimize
  - some are more succcint
- (Original) Codd's Theorem shows that the most important of the above languages, the so-called domain-independent, relational calculus and relational algebra, are equally expressive.
- By a set of translations we prove here a stronger result. The following languages are equally expressive:
  - domain-independent relational calculus,
  - relational calculus under the active domain semantics,
  - range-restricted relational calculus,
  - relational algebra,
  - non-recursive Datalog with negation,

Proof Strategy



Domain Independence

Some queries are "unsafe" and must be avoided:

- Domain Independence: Given a query and a database, the query must evaluate to the same result on the database no matter what the domain is assumed to be.
- Idea: exclude "unsafe" queries, i.e., in particular, queries that may yield an infinite answer.
- Let $Q_B(\mathcal{A})$ denote the result of evaluating query Q on database $\mathcal{A}$ assuming domain B.

Definition (domain-independence)

A query Q is *domain-independent* iff there do **not** exist

- a database instance $\mathcal{A}$ and
- two sets B, C that contain all constants that appear in $\mathcal{A}$ or in Q (also known as the active domain),

such that $Q_B(\mathcal{A}) \neq Q_C(\mathcal{A})$.

Queries Violating Domain Independence

Example (Unsafe Queries)

- $\{x \mid \neg R(x)\}$
  - $R = \emptyset, 1 \in B, 1 \notin C$: $1 \in Q_B(R), 1 \notin Q_C(R)$.
- $\{x \mid R(x) \vee R(y)\}$
- $\{y \mid R(x)\}$
- $\{x \mid R(x) \vee \neg R(x)\}$

Remark. Over infinite domains, these queries may yield an infinite result.

Undecidability of Domain Independence

Definition

For a domain-independent query Q and a database $\mathcal{A}$, we may define $Q(\mathcal{A}) := Q_B(\mathcal{A})$ for an arbitrary domain B that contains the active domain (the choice is irrelevant due to domain-independence).

- We would like to require domain-independence in queries.
- Domain-independence is an undecidable property for FO queries.
- Possible solutions:
  - semantic restriction on quantification
  - syntactic restriction on queries

Theorem

For every relational algebra expression there exists a domain-independent, relational calculus query.

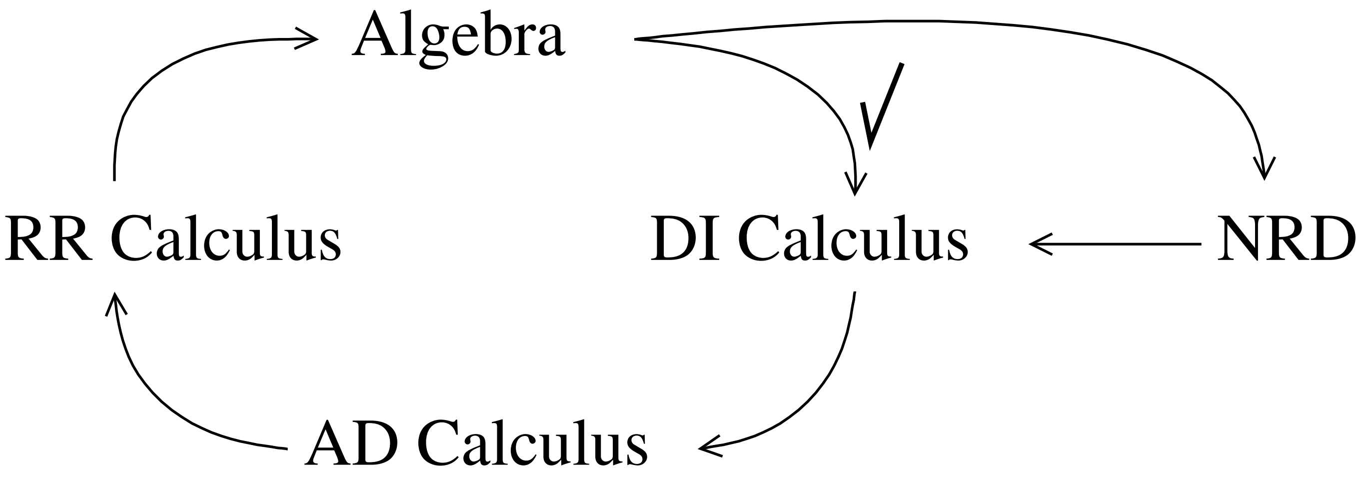
Proof.

Almost by definition:

$$
\begin{aligned}
R &:= \{\vec{x} \mid R(\vec{x})\} \\
\sigma_{x=y}(\{\vec{x} \mid \varphi(\vec{x})\}) &:= \{\vec{x} \mid \varphi(\vec{x}) \wedge x = y\} \\
\pi_{\vec{y}}(\{\vec{x} \mid \varphi(\vec{x})\}) &:= \{\vec{y} \mid \exists \vec{z}\, \varphi(\vec{x})\} \quad (\vec{x} = \vec{y}\vec{z}) \\
\{\vec{x} \mid \varphi(\vec{x})\} \times \{\vec{y} \mid \psi(\vec{y})\} &:= \{\vec{x}\vec{y} \mid \varphi(\vec{x}) \wedge \psi(\vec{y})\} \\
\{\vec{x} \mid \varphi(\vec{x})\} \cup \{\vec{x} \mid \psi(\vec{x})\} &:= \{\vec{x} \mid \varphi(\vec{x}) \vee \psi(\vec{x})\} \\
\{\vec{x} \mid \varphi(\vec{x})\} - \{\vec{x} \mid \psi(\vec{x})\} &:= \{\vec{x} \mid \varphi(\vec{x}) \wedge \neg\psi(\vec{x})\}
\end{aligned}
$$

It can be shown by an easy induction argument that the resulting relational calculus query is domain-independent. □

Current Status



Active Domain Interpretation

Definition

Active domain semantics for relational calculus query q over DB $\mathcal{A}$: variables range over the active domain, i.e. over values occurring in the database $\mathcal{A}$ and the query q, denoted by $adom(q, \mathcal{A})$.

Example

- Assume a database $\mathcal{A}$ with $R = \{\langle 1, 1 \rangle\}$ and $dom(R) = \{1, 2\}$.
- Consider the Boolean query $q = \{\langle\rangle \mid \forall x, y.R(x, y).\}$:
  - q is false in $\mathcal{A}$ under the standard semantics, but
  - q is true in $\mathcal{A}$ under the active domain semantics.

Theorem

For every domain-independent relational calculus query there is an equivalent relational calculus query under the active domain semantics.

Proof.

Assume a domain-independent query Q. We claim that Q itself is the desired query, such that its evaluation under standard semantics and under active domain semantics coincides.

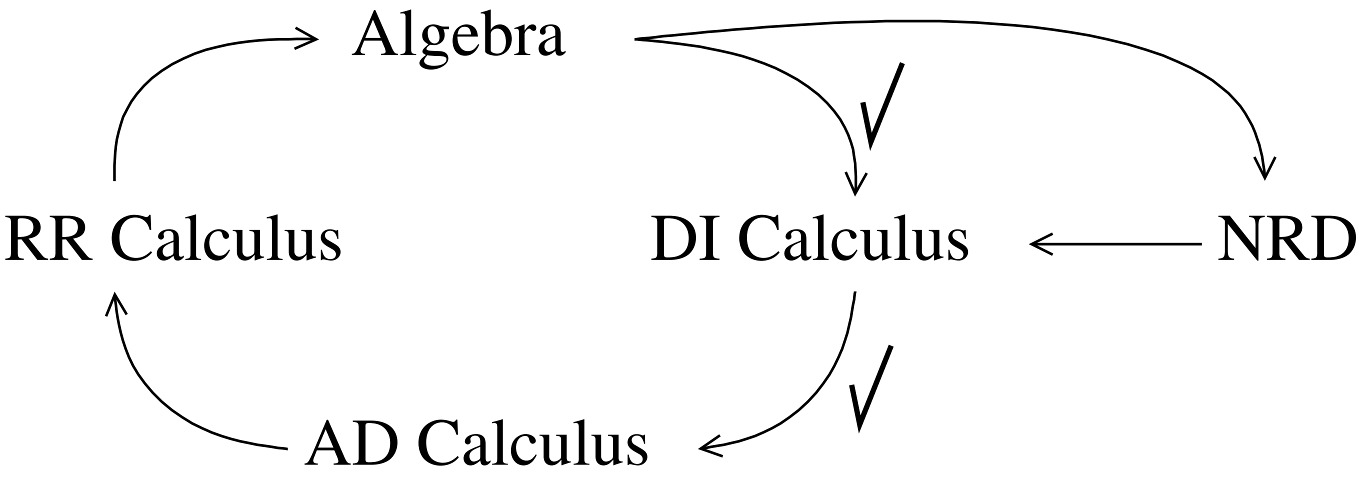
Consider an arbitrary database $\mathcal{A}$.

By domain-independence, $Q_B(\mathcal{A}) = Q_{adom(Q,\mathcal{A})}(\mathcal{A})$ under standard semantics for every domain B with $adom(Q, \mathcal{A}) \subseteq B$.

Likewise, under active domain semantics, $Q_B(\mathcal{A}) = Q_{adom(Q,\mathcal{A})}(\mathcal{A})$ for every domain B with $adom(Q, \mathcal{A}) \subseteq B$.

Finally, $Q_{adom(Q,\mathcal{A})}(\mathcal{A})$ yields the same result under standard semantics and under active domain semantics. Hence, for arbitary domain B with $adom(Q, \mathcal{A}) \subseteq B$, the set $Q_B(\mathcal{A})$ is the same under standard semantics and under active domain semantics. □

Current Status



Range-restricted Queries: a sufficient condition for DI

Preprocessing. A formula is turned into safe-range normal form (SRNF) by the following steps:

1. Variable substitution: no distinct pair of quantifiers may employ the same variable and no variable may occur both bound and free.
   Example: $(\exists x\ \varphi(x)) \vee (\exists x\ \psi(x)) \vdash (\exists x\ \varphi(x)) \vee (\exists x'\ \psi(x'))$.
2. Remove universal quantifiers: $\forall x\ \varphi \vdash \neg\exists x\ \neg\varphi$.
3. Remove implications: $\varphi \Rightarrow \psi \vdash \neg\varphi \vee \psi$.
4. Remove double negation: $\neg\neg\varphi \vdash \varphi$.
5. Flatten and/or, e.g.: $(\varphi \wedge \psi) \wedge \pi \vdash \varphi \wedge \psi \wedge \pi$.

Range-restriction Check

Definition

Given: Input formula π in SRNF.

$$
\begin{aligned}
rr(x = a) &:= \{x\} \\
rr(R(t_1, \ldots, t_n)) &:= Vars(\{t_1, \ldots, t_n\}) \\
rr(\varphi \land \psi) &:= rr(\varphi) \cup rr(\psi) \\
rr(\varphi \lor \psi) &:= rr(\varphi) \cap rr(\psi) \\
rr(\varphi \land x = y) &:= \begin{cases} rr(\varphi) & \ldots \quad \{x, y\} \cap rr(\varphi) = \emptyset \\ rr(\varphi) \cup \{x, y\} & \ldots \quad \text{otherwise} \end{cases} \\
rr(\neg\varphi) &:= \emptyset \\
rr(\exists x\, \varphi) &:= \begin{cases} rr(\varphi) - \{x\} & \ldots \quad x \in rr(\varphi) \\ \text{fail} & \ldots \quad \text{otherwise} \end{cases}
\end{aligned}
$$

If free(π) $\subseteq rr(\pi)$, then π is range-restricted (RR).

Range-restriction Examples

Example (in SRNF)

$$\overbrace{\exists z : \overbrace{\overbrace{P(x,y,z)}^{rr(\cdot)=\{x,y,z\}} \lor \overbrace{(\overbrace{R(x,y)}^{rr(\cdot)=\{x,y\}} \land \overbrace{(\overbrace{(\overbrace{S(z)}^{rr(\cdot)=\{z\}} \land \overbrace{\neg T(x,z)}^{rr(\cdot)=\emptyset})}^{rr(\cdot)=\{z\}} \lor T(y,z))}^{rr(\cdot)=\{z\}})}^{rr(\cdot)=\{x,y,z\}}}^{rr(\cdot)=\{x,y,z\}}}^{rr(\cdot)=\{x,y\}}$$

$rr(*) = \text{free}(*) = \{x, y\} \Rightarrow$ range-restricted!

Range-restricted Queries Capture AD Calculus

Theorem

For every relational calculus query under the active domain semantics there is an equivalent range-restricted relational calculus query.

Preparation of the proof.

Idea.

- define a predicate D that captures exactly the active domain,
- use it to effectively restrict the quantification to the active domain (relativization)

For an n-ary predicate R, let $dom(R, i)$ denote the formula
$\exists y_1, \ldots, \exists y_{i-1}, \exists y_{i+1}, \ldots, \exists y_n R(y_1, \ldots, y_{i-1}, x, y_{i+1}, \ldots, \exists y_n)$.
Let $D(x)$ be the disjunction of

- $dom(R, i)$ for all n-ary predicates of the schema and all $1 \leq i \leq n$, and
- $x = a$ for every constant of φ.

Range-restricted Queries Capture AD Calculus (continued)

Proof.

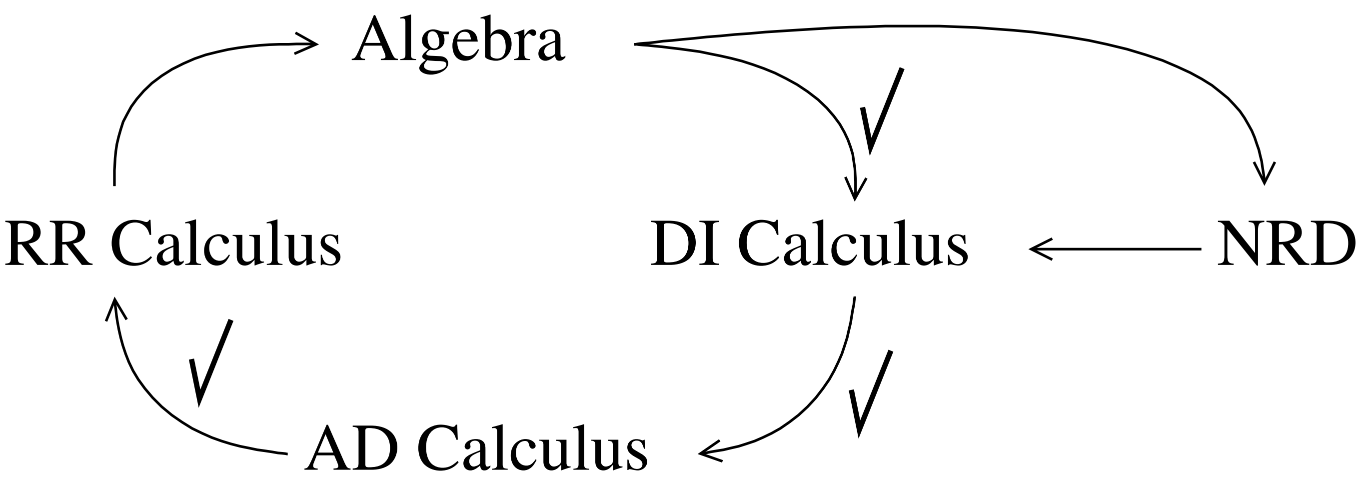
We now translate φ into an rr query φ' such that answering φ under the active domain semantics is equivalent to evaluating φ' under the standard semantics. (We write $tr(\psi)$ to denote the translation of a formula ψ.)

1. Turn φ into SRNF.
2. We build φ' from φ as follows:

$$\begin{aligned}
\varphi' &:= D(x_1) \wedge \ldots \wedge D(x_n) \wedge tr(\varphi), \\
&\qquad\qquad \text{where } \{x_1, \ldots, x_n\} = free(\varphi) \text{ and} \\
tr(A) &:= A \qquad (A \text{ is an atom}) \\
tr(\varphi \wedge \psi) &:= tr(\varphi) \wedge tr(\psi) \\
tr(\varphi \vee \psi) &:= tr(\varphi) \vee tr(\psi) \\
tr(\neg\varphi) &:= \neg tr(\varphi) \\
tr(\exists x\, \varphi) &:= (\exists x)\, (D(x) \wedge tr(\varphi))
\end{aligned}$$

□

Current Status



Theorem

For every range-restricted relational calculus query there exists an equivalent relational algebra expression.

Proof idea.

1. Put the query into SRNF.
2. Put the query into Relational-Algebra NF (RANF):
   - RANF: Each subformula is range-restricted. (Exception: In a subformula $\pi = \varphi_1 \wedge \cdots \wedge \varphi_k \wedge \neg\psi$, π has to be RR and the φ_i and ψ have to be in RANF, but $\neg\psi$ does not need to be RR).
3. Translate the RANF formula into relational algebra. This can be done inductively from the leaves to the root of the parse tree of the formula.

From SRNF to RANF

Given a formula in SRNF.

- The formula is in RANF if each subformula is range-restricted.
- Possible obstacles: subformulae of the form $\varphi \vee \psi$ or $\neg\varphi$.
- Only these remove possibly relevant variables from rr.
- Solution: relativize using the active domain relation D:

$$\varphi \vee \psi \quad \vdash \quad \underbrace{(\varphi \wedge \bigwedge_{x \in (\mathrm{free}(\psi) - \mathrm{free}(\varphi))} D(x)) \vee (\psi \wedge \bigwedge_{x \in (\mathrm{free}(\varphi) - \mathrm{free}(\psi))} D(x))}_{rr(*) = \mathrm{free}(*) = \mathrm{free}(\varphi) \cup \mathrm{free}(\psi)}$$

$$\neg\varphi \quad \vdash \quad (\neg\varphi \wedge \bigwedge_{x \in \mathrm{free}(\varphi)} D(x))$$

- Shorter (=better) RANF queries can be achieved by rewriting the input formula using equivalences we already know.

From RANF to Relational Algebra

RANF formulae can be translated to relational algebra using the following rules:

$$
\begin{array}{rcl}
Alg(\varphi \land \psi) & := & Alg(\varphi) \bowtie Alg(\psi) \\
Alg(\varphi \land \neg\psi) & := & Alg(\varphi) - Alg(\psi) \\
 & \ldots & \varphi \text{ and } \psi \text{ have the same schema} \\
Alg(\exists y\ \varphi(\vec{x}, y)) & := & \pi_{\vec{x}} Alg(\varphi(\vec{x}, y)) \\
Alg(\varphi \land x\vartheta t) & := & \sigma_{x\vartheta t} Alg(\varphi) \\
 & \ldots & \vartheta \text{ is either } = \text{ or } \neq \text{ and } t \text{ is a term.} \\
Alg(R(x_1, \ldots, x_k)) & := & \rho_{A_1 \ldots A_k \to x_1 \ldots x_k} R \\
 & \ldots & \text{relation } R \text{ has schema } R(A_1, \ldots, A_k)
\end{array}
$$

From RR Queries to the Algebra

Example

Let D be the active domain relation with schema $D(D)$ and let R have schema $R(A, B)$. The formula

$$\exists x \left(D(x) \land \exists y\ (D(y) \land \neg R(x, y))\right)$$

corresponds to the RANF formula

$$\exists x\ \exists y \left((D(x) \land D(y)) \land \neg R(x, y)\right).$$

An equivalent relational algebra expression looks as follows.

$$\begin{aligned} & Alg(\exists x\ \exists y\ ((D(x) \land D(y)) \land \neg R(x, y))) \\ \vdash\quad & \pi_\emptyset\left(Alg((D(x) \land D(y)) - Alg(R(x, y))\right) \\ \vdash\quad & \pi_\emptyset((\rho_x D \bowtie \rho_y D) - \rho_{xy} R) \end{aligned}$$

From RR Queries to the Algebra

Example

Range-restricted but not in RANF (i.e., not locally range-restricted):

$$\{(x,y) \mid \exists z : P(x,y,z) \lor (R(x,y) \land (\overbrace{(S(z) \land \neg T(x,z))}^{rr(*)=\{z\}} \lor T(y,z)))\}$$

We transform this formula using the rewrite rule

$$\varphi \land (\psi_1 \lor \psi_2) \vdash (\varphi \land \psi_1) \lor (\varphi \land \psi_2)$$

into RANF:

$$\{(x,y) \mid \exists z : P(x,y,z) \lor (R(x,y) \land S(z) \land \neg T(x,z)) \lor \\ (R(x,y) \land T(y,z))\}$$

From RR Queries to the Algebra

Example (in RANF)

$$\{(x,y) \mid \exists z : \overbrace{P(x,y,z)}^{e_1=\rho_{xyz}P} \vee \overbrace{\overbrace{(\overbrace{R(x,y)}^{\rho_{xy}R} \wedge \overbrace{S(z)}^{\rho_z S}}^{(\cdot)\bowtie(\cdot)} \wedge \neg \overbrace{T(x,z)}^{e_{21}=\rho_{xz}T})}^{e_2=(\cdot)-((\rho_y D)\bowtie(\cdot))} \vee \overbrace{(\overbrace{R(x,y)}^{\rho_{xy}R} \wedge \overbrace{T(y,z)}^{\rho_{yz}T})}^{e_3=(\cdot)\bowtie(\cdot)}\}$$

Equivalent relational algebra expression: $\pi_{xy}(e_1 \cup e_2 \cup e_3)$.

From RR Queries to the Algebra

Example

"Select all professors (P) who only give lectures (L) in the field of computer science (C)." The schemata are $P(P), L(P, C), C(C)$ and the active domain is given by a relation D with schema $D(D)$.

$$\{x \mid P(x) \land \forall y(L(x,y) \rightarrow C(y))\}$$

$$\overset{\text{to SRNF}}{\vdash} \quad \{x \mid P(x) \land \neg\exists y(L(x,y) \land \neg C(y))\}$$

$$\overset{\text{to RANF}}{\vdash} \quad \underbrace{\{x \mid P(x) \land \neg\, \underbrace{\exists y(\underbrace{L(x,y) \land (\underbrace{D(y) \land \neg C(y)}_{(\rho_C D)-C})}_{L \bowtie ((\rho_C D)-C)})}_{\pi_P(L \bowtie ((\rho_C D)-C))})\}}_{P-\pi_P(L \bowtie ((\rho_C D)-C))}$$

From RR Queries to the Algebra

Example

The schemata are $L(P, C), C(C)$ and the domain is given by $D(D)$.

$$\{\langle x, y\rangle \mid L(x,y) \wedge \neg C(y)\} \equiv \{\langle x, y\rangle \mid \underbrace{L(x,y) \wedge (D(y) \wedge \neg C(y))}_{L \bowtie ((\rho_C D) - C)}\}$$

$$\equiv \{\langle x, y\rangle \mid \underbrace{L(x,y) \wedge \neg(D(x) \wedge C(y))}_{L - ((\rho_C D) \times C)}\}$$

D	D
	1
	2
	3
	4

L	P	C
	1	2
	3	4

C	C
	2

$(\rho_C D) - C$	C
	1
	3
	4

$L \bowtie ((\rho_C D) - C)$	P	C
	3	4

$(\rho_C D) \times C$	P	C
	1	2
	2	2
	3	2
	4	2

From RR Queries to the Algebra

Example

Schema $R(AB)$, $S(C)$, $T(AC)$; active domain $D(D)$.

$$
\begin{aligned}
& \{\langle x, y, z\rangle \mid R(x, y) \land (S(z) \land \neg T(x, z))\} \\
\vdash \quad & \{\langle x, y, z\rangle \mid \underbrace{R(x, y) \land \underbrace{(\underbrace{(D(x) \land S(z))}_{D \times S} \land \neg T(x, z))}_{(D\times S)-T}}_{R \bowtie ((D\times S)-T)}\}
\end{aligned}
$$

From RR Queries to the Algebra

Example

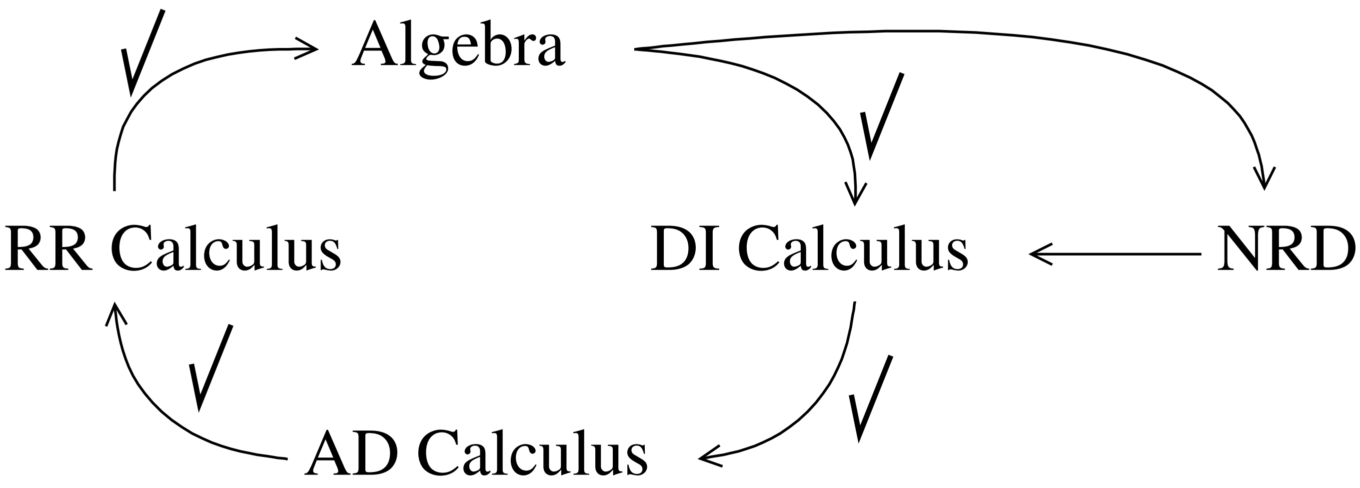
Schema $R(AB)$, $S(AC)$, $T(BC)$; active domain $D(D)$.

$$\begin{aligned} & \{\langle x,y,z\rangle \mid R(x,y) \wedge (S(x,z) \vee T(y,z))\} \\ \vdash \quad & \{\langle x,y,z\rangle \mid \underbrace{(\underbrace{R(x,y) \wedge S(x,z)}_{R \bowtie S}) \vee (\underbrace{R(x,y) \wedge T(y,z)}_{R \bowtie T})}_{(R \bowtie S) \cup (R \bowtie T)}\} \end{aligned}$$

$$\begin{aligned} \text{or} \quad & \{\langle x,y,z\rangle \mid R(x,y) \wedge (S(x,z) \vee T(y,z))\} \\ \vdash \quad & \{\langle x,y,z\rangle \mid \underbrace{R(x,y) \wedge ((S(x,z) \wedge D(y)) \vee (T(y,z) \wedge D(x)))}_{R \bowtie ((S \times \rho_B D) \cup (T \times \rho_A D))}\} \end{aligned}$$

This is correct because
$R(x,y) \wedge (\varphi \vee \psi) \equiv R(x,y) \wedge D(x) \wedge D(y) \wedge (\varphi \vee \psi) \equiv$
$R(x,y) \wedge ((D(x) \wedge D(y) \wedge \varphi) \vee (D(x) \wedge D(y) \wedge \psi))$.

Current Status



Non-recursive Datalog with negation

Definition

Non-recursive Datalog with negation (nr-$Datalog^{\neg}$) prohibits cycles in the dependency graph $DEP(P)$ of a program P.

- non-recursiveness means that negation is trivially stratified!

Theorem

For every relational algebra query there exists an equivalent nr-$Datalog^{\neg}$ query.

Proof idea.

- Inductively, for each algebra expression E, we construct the program P_E that defines the predicate $\mathtt{Q}_E$ of the same arity as E.
- We proceed by a bottom-up traversal of the syntax tree of algebra expression E, introducing a new predicate for each subexpression of E. Hence, the resulting program is clearly non-recursive.

From Algebra to nr-*Datalog*$^\neg$

Proof.

$$P_E := \{\mathtt{Q}_E(\vec{x}) \;\mathtt{:-}\; \mathtt{R}(\vec{x})\} \quad \text{if } E \text{ is a relation } R$$

$$P_{\sigma_{x=y}(E)} := \{\mathtt{Q}_{\sigma_{x=y}(E)}(\vec{x}) \;\mathtt{:-}\; \mathtt{Q}_E(\vec{x}) \land x = y\} \cup P_E$$

$$P_{\pi_{\vec{y}}(E)} := \{\mathtt{Q}_{\pi_{\vec{y}}(E)}(\vec{y}) \;\mathtt{:-}\; \mathtt{Q}_E(\vec{x})\} \cup P_E$$

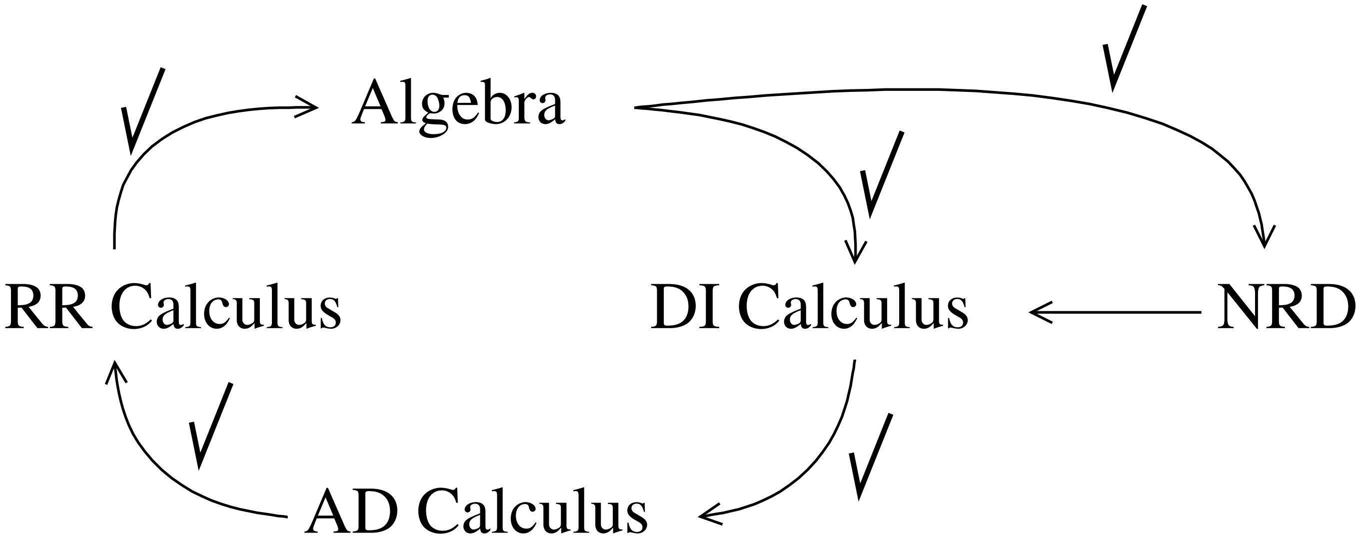
$$P_{E_1 \times E_2} := \{\mathtt{Q}_{E_1 \times E_2}(\vec{x}, \vec{y}) \;\mathtt{:-}\; \mathtt{Q}_{E_1}(\vec{x}), \mathtt{Q}_{E_2}(\vec{y})\} \cup P_{E_1} \cup P_{E_2}$$

$$P_{E_1 \cup E_2} := \{\mathtt{Q}_{E_1 \cup E_2}(\vec{x}) \;\mathtt{:-}\; \mathtt{Q}_{E_1}(\vec{x}))\} \cup \{\mathtt{Q}_{E_1 \cup E_2}(\vec{x}) \;\mathtt{:-}\; \mathtt{Q}_{E_2}(\vec{x}))\} \cup P_{E_1} \cup P_{E_2}$$

$$P_{E_1 - E_2} := \{\mathtt{Q}_{E_1 - E_2}(\vec{x}) \;\mathtt{:-}\; \mathtt{Q}_{E_1}(\vec{x}), \mathtt{not}\ \mathtt{Q}_{E_2}(\vec{x})\} \cup P_{E_1} \cup P_{E_2}$$

Clearly, E and P_E have the same answer, i.e. for any database DB and any constant tuple $\vec{c}$, we have $\vec{c} \in M[E](DB)$ iff $P_E^* \land DB^* \models \mathtt{Q}_E(\vec{c})$. □

Current Status



From nr-*Datalog*$^\neg$ to DI Calculus

Theorem

For every query in non-recursive Datalog with negation there exists an equivalent domain-independent relational calculus query.

Proof idea.

- Let us assume an nr-*Datalog*$^\neg$ program P.
- For every predicate R occurring in P we define a formula $\varphi_{P,R}$.
- $\varphi_{P,R}$ captures the query defined by the program P "restricted to predicate R", i.e., for any database DB the following are equal:
  - the answer to $\{\vec{x} \mid \varphi_{P,R}(\vec{x})\}$, where $\vec{x}$ are the free variables of $\varphi_{P,R}$;
  - the set of constant tuples $\vec{c}$ with $P^* \wedge DB^* \models R(\vec{c})$.

From nr-*Datalog*$^{\neg}$ to DI Calculus (continued)

Proof

W.l.o.g., we assume:

- P has no constants and no multiple occurrences of variables in rule heads: this can be simulated using (fresh) variables and $=$.
- for every pair $H_1(\vec{x}_1)$, $H_2(\vec{x}_2)$ of head atoms in P, $H_1 = H_2$ implies $\vec{x}_1 = \vec{x}_2$ (can be achieved by variable renaming)

Inductive definition of $\varphi_{P,R}$:

- (Base case) If R is a predicate that does not appear in the head of any rule in P, then $\varphi_{P,R} = R(\vec{x})$.
- (Inductive step) Choose a predicate Q in P, s.t. for all predicates W occurring in the body of some rule with head predicate Q, the formula $\varphi_{P,W}$ has already been defined.

Proof.

Suppose that the following are the rules with head predicate Q:

$$\begin{array}{lll} \mathtt{Q}(\vec{x}) & :- & \alpha_{0,0}, \ldots, \alpha_{0,n_0} \\ & \vdots & \\ \mathtt{Q}(\vec{x}) & :- & \alpha_{m,0}, \ldots, \alpha_{m,n_m} \end{array}$$

Then we set

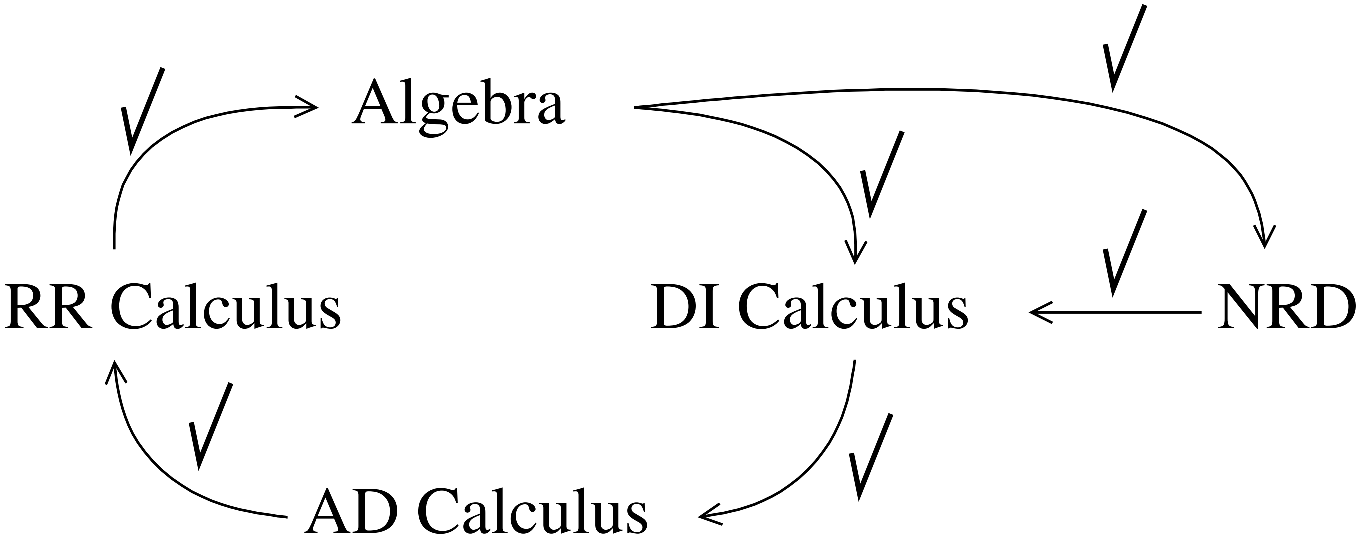
$$\varphi_{P,Q} = \bigvee_{i \in \{0,\ldots,m\}} (\exists \vec{y}_i : \bigwedge_{j \in \{0,\ldots,n_i\}} \beta_{i,j}),$$

where $\vec{y}_i$ are the existential variables of the ith rule (i.e. the variables not occurring in the rule head), and

$$\beta_{i,j} = \begin{cases} \varphi_{P,W}(v) & \text{if } \alpha_{i,j} \text{ is an atom } \mathtt{W}(v). \\ \neg\varphi_{P,W}(v) & \text{if } \alpha_{i,j} \text{ is an atom } \mathtt{not}\ \mathtt{W}(v). \end{cases}$$

□

Current Status



Aggregation-free SQL

Aggregation-free SQL is just a different format for expressing relational algebra queries:

Theorem

Relational algebra and aggregation-free SQL queries are equally expressive.

Learning objectives

Understanding:

- the notion of domain-independence,
- the active domain semantics,
- the notion of range restricted queries,
- Codd's Theorem: equivalence of various relational query languages in terms expressive power.

More details: Serge Abiteboul, Richard Hull, Victor Vianu: Foundations of Databases. Chapter 5. Addison-Wesley 1995,