A Retrospective on Naturally Embedded Query Languages

Peter Buneman, Val Tannen,

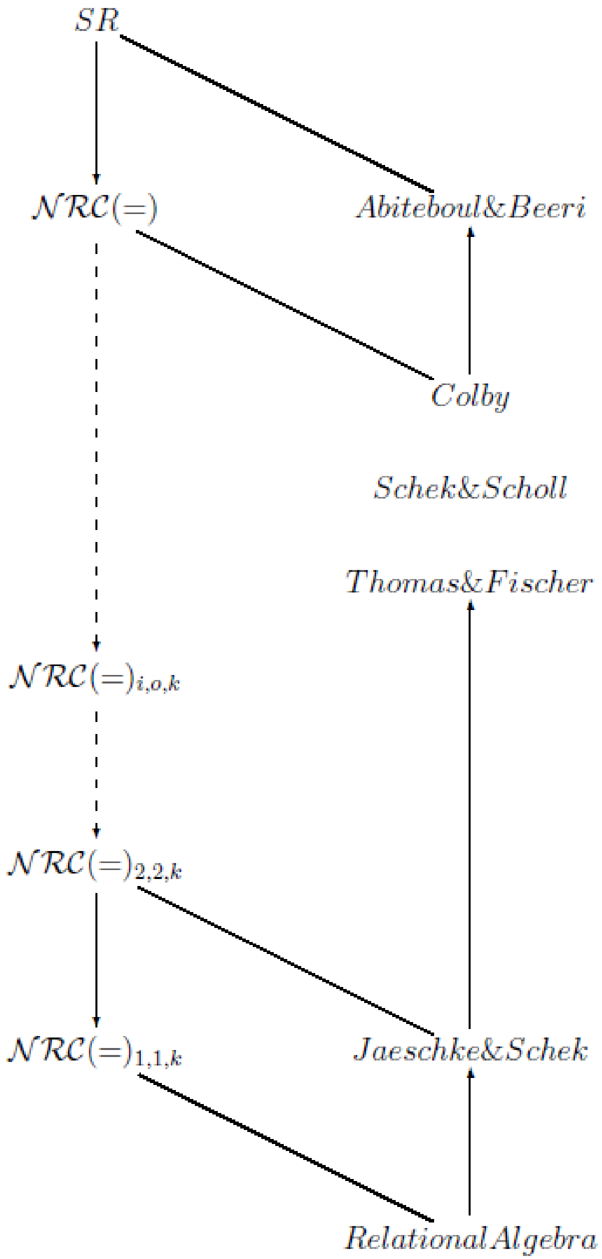
Limsoon Wong

NUS National University of Singapore

Outline

- **Design of query languages**
- **Engineering data integration systems**
- **Understanding expressive power**
- **Exploring intensional expressive power**
- **Adding annotations**
- **Open problems**

DESIGN OF QUERY LANGUAGES



Two ways to develop query languages

Structural Recursion

- **Let $u : t \times t \rightarrow t$, $f : s \rightarrow t$, and $e : t$ be such that $\langle t, u, e \rangle$ forms a commutative idempotent monoid. Then there is a unique $h : \{s\} \rightarrow t$ satisfying**

$$\begin{aligned} h\{\} &= e \\ h\{x\} &= f(x) \\ h(R \cup S) &= u(h(R),\ h(S)) \end{aligned}$$

- **Such a *h* is said to be defined by structural recursion on the union representation of sets. Denote this *h* by *sru(u, f, e)***

MapReduce is Structural Recursion

- ***sru(u, f, e) {o1, …, on} = f(o_1) u … u f(o_n) u e***

- **The function *f* is "map"; it is applied (in parallel) to all elements in the input set**

- **The function *u* is "reduce"; it is applied (in parallel) to combine the results of the map**

Examples

- **Structural recursion is expressive and can be used to write relatively efficient queries**

$$cartprod(R, S) \triangleq sru(\cup,\ \lambda x.sru(\cup,\ \lambda y.\{(x, y)\},\ \{\})(S),\ \{\})(R)$$

$$map(f) \triangleq sru(\cup,\ \lambda x.\{f(x)\},\ \{\})$$

$$flatten \triangleq sru(\cup,\ \lambda x.x,\ \{\})$$

$$powset \triangleq sru(flatten \circ map(\cup) \circ cartprod,\ \lambda x.\{\{\},\ \{x\}\}, \{\{\}\})$$

- **But $\langle t, u, e \rangle$ has to be a commutative idempotent monoid in order for *sru(u, f, e)* to be well defined on sets. E.g., *sru(+, λx.1, 0)* is not well defined**

⇒ Restrict use of structural recursion to *sru(∪, f, {})*, which is always well defined

More considerations in (Tannen, Subrahmanyam, ICALP91)

Nested Relational Calculus (NRC)

- **Types**

$$s, t ::= \ | \ bool \ | \ b \ | \ s \times t \ | \ \{s\}$$

- **Expressions**

$$\frac{}{x^s : s} \qquad \frac{e_1 : s \quad e_2 : t}{(e_1, e_2) : s \times t} \qquad \frac{e : s \times t}{\pi_1 \ e : s \quad \pi_2 \ e : t}$$

$$\frac{}{true \ : bool} \qquad \frac{}{false \ : bool} \qquad \frac{e_1 : bool \quad e_2 : s \quad e_3 : s}{if \ e_1 \ then \ e_2 \ else \ e_3 : s}$$

$$\frac{}{\{\}^s : \{s\}} \qquad \frac{e : s}{\{e\} : \{s\}} \qquad \frac{e_1 : \{s\} \quad e_2 : \{s\}}{e_1 \bigcup e_2 : \{s\}}$$

$$\frac{e_1 : \{s\} \quad e_2 : \{t\}}{\bigcup\{e_1 \mid x^t \in e_2\} : \{s\}} \qquad \frac{e : \{s\}}{empty \ e : bool} \qquad \frac{e_1 : s \quad e_2 : s}{e_1 = e_2 : bool}$$

where $\cup\{e_1 \mid x \in e_2\} = sru(\cup, \lambda x.e1, \{\})(e2)$

NRC is equivalent to …

- **These operations are expressible in NRC: Project, Join, Union, Select, Difference, Intersect, Unnest, Nest. E.g.:**

  - **Relational projection**

    $\Pi_2(R) := \bigcup\{\{\pi_2\, x\} \mid x \in R\}$

  - **Relational selection**

    $\sigma(p)(R) := \bigcup\{\text{if } p(x) \text{ then } \{x\} \text{ else } \{\} \mid x \in R\}$

  - **Cartesian product**

    $\otimes(R,S) := \bigcup\{\bigcup\{\{(x,y)\} \mid x \in R\} \mid y \in S\}$

- Theorem 1 (Tannen, Buneman, Wong, ICDT92)

  NRC has the same expressive power as the algebras of Schek&Scholl, Thomas&Fischer, etc.

Comprehension Syntax

- **Translating into comprehension syntax**

$$\bigcup\{e_1 \mid x \in e_2\} = \{ y \mid x \in e_2, y \in e_1\}$$

- **Translating from comprehension syntax**

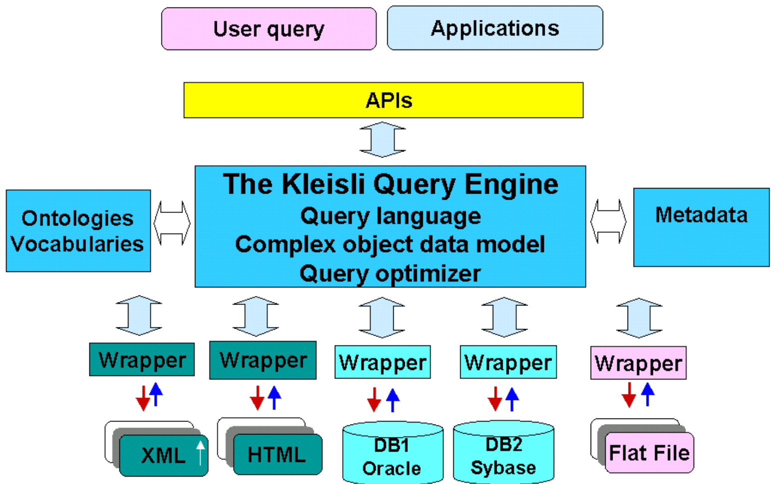
$$\{ e_1 \mid x \in e_2, \Delta \} = \bigcup\{ \{e_1 \mid \Delta \} \mid x \in e_2\}$$
$$\{ e_1 \mid C, \Delta\} = \text{if } C \text{ then } \{e_1 \mid \Delta \} \text{ else } \{ \}$$
$$\{ e_1 \mid \} = \{ e_1 \}$$

$\Rightarrow$ **Treat comprehension as a nice syntactic sugar**

Further articulation in (Buneman, Libkin, Suciu, Tannen, Wong, SIGMOD Record 94)

ENGINEERING DATA INTEGRATION SYSTEMS

Kleisli Query System



Buneman, Davidson, Hart, Overton, Wong, VLDB95
Wong, ICFP00

- **Nested set/bag/list model**
- **Self-describing data exchange format**
- **Lots of thin wrappers**
- **High-level query language with type inference**
- **Powerful query optimizer**
- **Nested set/bag/list store**

US DOE "Impossible Query", 1993

- **For each gene on a given cytogenetic band, find its non-human homologs**

source	type	location	remarks
GDB	Sybase	Baltimore	Flat tables SQL joins Location info
Entrez	ASN.1	Bethesda	Nested tables Keywords Homolog info

Solution in Kleisli

- **Using Kleisli:**
  - Clear
  - Succinct
  - Efficient
- **Handles**
  - Heterogeneity
  - Complexity

```
sybase-add (#name:”GDB", ...);
create view  L from locus_cyto_location using GDB;
create view E from object_genbank_eref using GDB;
select
    #accn: g.#genbank_ref,   #nonhuman-homologs: H
from
    L as c,  E as g,
    {select u
     from g.#genbank_ref.na-get-homolog-summary as u
     where not(u.#title string-islike "%Human%") &
               not(u.#title string-islike "%H.sapien%")} as H
where
    c.#chrom_num = "22” &
    g.#object_id = c.#locus_id &
    not (H = { });
```

UNDERSTANDING EXPRESSIVE POWER

Conservative Extension Property

A language $\mathcal{L}$ has conservative extension property if

for every function f definable in $\mathcal{L}$,
there is an implementation f^* of f in $\mathcal{L}$ such that

for any input i and corresponding output o,
each intermediate data item created in the course of executing f^* on i to produce o has set nesting complexity no more than that of i and o

Expressive Power of NRC

- Theorem 2 (Wong, PODS93)

  NRC has the conservative extension property

- Corollary 3

  Every function from flat relations to flat relations expressible in NRC is expressible in relational algebra

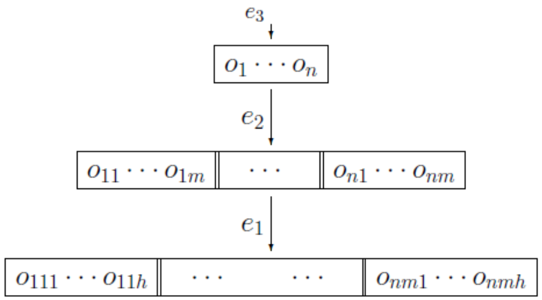
Proof Idea

- **Strongly normalizing rewrite system**

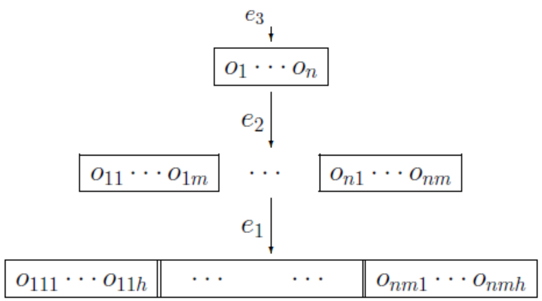
- $(\lambda x.e)(e') \rightsquigarrow e[e'/x]$
- $\pi_i(e_1, e_2) \rightsquigarrow e_i$
- $e\ (\mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3) \rightsquigarrow \mathit{if}\ e_1\ \mathit{then}\ e\ e_2\ \mathit{else}\ e\ e_3$
- $\mathit{if}\ \mathit{true}\ \mathit{then}\ e_2\ \mathit{else}\ e_3 \rightsquigarrow e_2$
- $\mathit{if}\ \mathit{false}\ \mathit{then}\ e_2\ \mathit{else}\ e_3 \rightsquigarrow e_3$
- $\bigcup\{e \mid x \in e_1 \cup e_2\} \rightsquigarrow \bigcup\{e \mid x \in e_1\} \cup \bigcup\{e \mid x \in e_2\}$
- $\bigcup\{e \mid x \in \{\}\} \rightsquigarrow \{\}$
- $\bigcup\{e \mid x \in \{e'\}\} \rightsquigarrow e[e'/x]$
- $\bigcup\{e \mid x \in \mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3\}$
  $\rightsquigarrow \mathit{if}\ e_1\ \mathit{then}\ \bigcup\{e \mid x \in e_2\}\ \mathit{else}\ \bigcup\{e \mid x \in e_3\}$
- $\bigcup\{e_1 \mid x \in \bigcup\{e_2 \mid y \in e_3\}\} \rightsquigarrow \bigcup\{\bigcup\{e_1 \mid x \in e_2\} \mid y \in e_3\}$

- **Vertical loop fusion**

$\bigcup\{e_1 \mid x \in \bigcup\{e_2 \mid y \in e_3\}\}$



$\rightsquigarrow \bigcup\{\bigcup\{e_1 \mid x \in e_2\} \mid y \in e_3\}$



Theoretical Reconstruction of SQL

- **Expressions of NRC(Q,+,•,–,÷,Σ,=, ≤^{Q}) are those of NRC plus the followings**

$$\frac{e_1 : \mathbb{Q} \quad e_2 : \mathbb{Q}}{e_1 + e_2 : \mathbb{Q}} \qquad \frac{e_1 : \mathbb{Q} \quad e_2 : \mathbb{Q}}{e_1 \cdot e_2 : \mathbb{Q}} \qquad \frac{e_1 : \mathbb{Q} \quad e_2 : \mathbb{Q}}{e_1 \div e_2 : \mathbb{Q}}$$

$$\frac{e_1 : \mathbb{Q} \quad e_2 : \mathbb{Q}}{e_1 - e_2 : \mathbb{Q}} \qquad \frac{e_1 : \mathbb{Q} \quad e_2 : \{s\}}{\Sigma\{\!|e_1 \mid x^s \in e_2|\!\} : \mathbb{Q}} \qquad \frac{e_1 : \mathbb{Q} \quad e_2 : \mathbb{Q}}{e_1 \leq e_2 : bool}$$

- **Here** $\Sigma\{| e_1 \mid x \in e_2 |\} = f(o_1) + \ldots + f(o_n)$**, where** f **is the function** $f(x) = e_1$ **and** $\{o_1, \ldots, o_n\}$ **is the set** e_2

Example Aggregate Functions

- **Count the number of records**

  $$count(R) := \Sigma\{\!|\ 1 \mid x \in R\ |\!\}$$

- **Total the first column**

  $$total_1(R) := \Sigma\{\!|\ \pi_1\, x \mid x \in R\ |\!\}$$

- **Average of the first column**

  $$ave_1(R) := total_1(R) \div count(R)$$

- **A totally generic query expressible in SQL but inexpressible in FO(=)**

  $$eqcard(R,S) := count(R) = count(S)$$

Expressive Power of NRC(Q,+,•,–,÷,Σ,=, $\leq^{Q}$)

- Theorem 4 (Libkin, Wong, DBPL93)

  NRC(Q,+,•,–,÷,Σ,=, $\leq^{Q}$) has the conservative extension property

- Corollary 5

  Every function from flat relations to flat relations is expressible in NRC(Q,+,•,–,÷,Σ,=, $\leq^{Q}$) iff it is also expressible in "entry-level" SQL

Finite/Co-finite Property I

- Theorem 6 (Libkin, Wong, DBPL93)

  Let *P : Q → B* be a predicate definable in NRC(Q,+,•,–,÷,Σ,=, $\leq^{Q}$). Then either *P* holds for finitely many natural numbers or *P* fails for finitely many natural numbers

- Corollary 7

  NRC(Q,+,•,–,÷,Σ,=, $\leq^{Q}$) cannot test whether a natural number is even or odd

Proof Idea

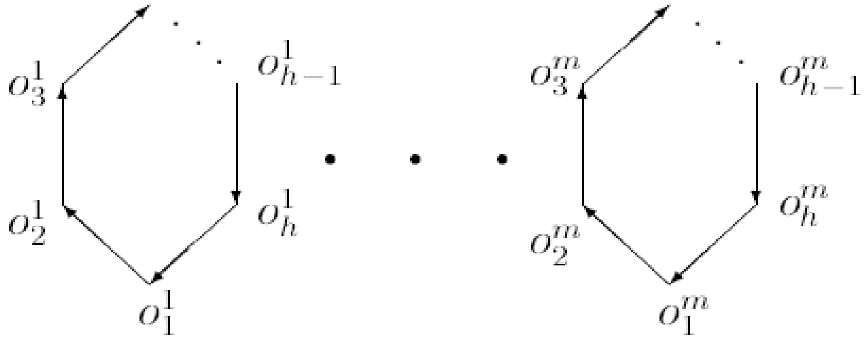
- ***P : Q → B*** **has height 0. By conservative extension property on NRC(Q,+,•,–,÷,Σ,=, $\leq^Q$), any implementation of it in NRC(Q,+,•,–,÷,Σ,=, $\leq^Q$) is equivalent to one that does not use sets. Such an implementation must be equivalent to something like**

$$\lambda n.(\bigvee \bigwedge p_i(n) = 0) \vee (\bigvee \bigwedge p_i(n) \neq 0)$$

where $p_i(n)$ are polynomials in n.

- **Finite/co-finiteness then follows from the fact that polynomials have finite number of roots**

Finite/Co-finite Property II



h-multi-cycle

- Theorem 8 (Libkin, Wong, PODS94)

  Let *P : {b × b} → B* be a predicate definable in NRC(Q,+,•,–,÷,Σ,=, ≤^{Q}). Then there is a *h* such that either *P* holds for all *h*-multi-cycles or *P* fails for all *h*-multi-cycles

- Corollary 9

  NRC(Q,+,•,–,÷,Σ,=, ≤^{Q}) cannot test the parity of a set and cannot express transitive closure

Locality Property

A language $\mathcal{L}$ has locality property if the result of every flat relational query f definable in $\mathcal{L}$ is determined by a small neighbourhood of its input

I.e., for all flat relational query expression *e[R]* in $\mathcal{L}$,

there is a finite number *r* such that,

for all $\mathcal{A} = \langle A,O \rangle$ in *STRUCT[R]*,

for all two m-ary vectors *a* and *b* of elements in $\mathcal{A}$,

$N_r^{\mathcal{A}}(a) \approx N_r^{\mathcal{A}}(b)$ implies

$a \in e[O/R]$ if and only if $b \in e[O/R]$

Notations: $N_r^{\mathcal{A}}(b)$ means the neighbourhood of *b* in $\mathcal{A}$, *up to a radius r.*

Bounded Degree Property

A language $\mathcal{L}$ has bounded degree property if

for every function f, on graphs, definable in $\mathcal{L}$, and for any number k,

there is a number c such that

for any graph $\mathcal{G}$ with deg($\mathcal{G}$) $\in$ { 0, 1, …, k},

it is the case that $c \geq$ card(deg(f($\mathcal{G}$)))

That is, $\mathcal{L}$ cannot define a function that produces complex graphs from simple graphs

Expressive Power of NRC(Q,+,•,–,÷,Σ,=, $\leq^Q$)

- Theorem 10 (Dong, Libkin, Wong, ICDT97)

  NRC(Q,+,•,–,÷,Σ,=, $\leq^Q$) has the locality property, when restricted to flat relational queries on input structures of degree less than some fixed k

- Theorem 11 (Dong, Libkin, Wong, ICDT97)

  Every language that has the locality property also has the bounded degree property

- Theorem 12 (Dong, Libkin, Wong, ICDT97)

  NRC(Q,+,•,–,÷,Σ,=, $\leq^Q$) has the bounded degree property

EXPLORING INTENSIONAL EXPRESSIVE POWER

What is intensional expressive power?

- **Saying a function with linear complexity is expressible in a given query language *is not* the same as saying its implementation in that query language has linear complexity**

I.e., we are looking at

- **What the *algorithms* expressible in a query language are,**
- **Rather than what the *functions* expressible in a query language are**

NRC(powerset)

$$\frac{}{c : b} \quad \frac{}{x^s : s} \quad \frac{e_1 : s_1 \quad \ldots \quad e_n : s_n}{(e_1, \ldots, e_n) : s_1 \times \cdots \times s_n} \quad \frac{e : s_1 \times \cdots \times s_n}{\pi_i\ e : s_i} 1 \leq i \leq n$$

$$\frac{}{\{\}^s : \{s\}} \quad \frac{e : s}{\{e\} : \{s\}} \quad \frac{e_1 : \{s\} \quad e_2 : \{s\}}{e_1 \cup e_2 : \{s\}} \quad \frac{e_1 : \{s\} \quad e_2 : \{t\}}{\bigcup\{e_1 \mid x^t \in e_2\} : \{s\}}$$

$$\frac{}{true : bool} \quad \frac{}{false : bool} \quad \frac{e_1 : bool \quad e_2 : s \quad e_3 : s}{if\ e_1\ then\ e_2\ else\ e_3 : s}$$

$$\frac{e_1 : b \quad e_2 : b}{e_1 = e_2 : bool} \quad \frac{e : \{b \times \cdots \times b\}}{isempty\ e : bool}$$

Powerset Operator in $\mathcal{NRC}(powerset)$

$$\frac{e : \{b \times \cdots \times b\}}{powerset\ e : \{\{b \times \cdots \times b\}\}}$$

NRC cannot express recursive queries. Adding a powerset operation enables this.

Operational Semantics

$$\frac{}{c \Downarrow c} \qquad \frac{e_1 \Downarrow C_1 \quad \ldots \quad e_n \Downarrow C_n}{(e_1, \ldots, e_n) \Downarrow (C_1, \ldots, C_n)} \qquad \frac{e \Downarrow (C_1, \ldots, C_n)}{\pi_i \; e \Downarrow C_i} 1 \leq i \leq n$$

$$\frac{}{\{\} \Downarrow \{\}} \qquad \frac{e \Downarrow C}{\{e\} \Downarrow \{C\}} \qquad \frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 \cup e_2 \Downarrow C_1 \cup C_2}$$

$$\frac{e_2 \Downarrow \{C_1, \ldots, C_n\} \quad e_1[C_1/x] \Downarrow C'_1 \quad \cdots \quad e_1[C_n/x] \Downarrow C'_n}{\bigcup\{e_1 \mid x \in e_2\} \Downarrow C'_1 \cup \cdots \cup \; C'_n}$$

$$\frac{}{true \Downarrow true} \qquad \frac{}{false \Downarrow false}$$

$$\frac{e_1 \Downarrow true \quad e_2 \Downarrow C}{if \;\; e_1 \;\; then \; e_2 \; else \; e_3 \Downarrow C} \qquad \frac{e_1 \Downarrow false \quad e_3 \Downarrow C}{if \;\; e_1 \;\; then \; e_2 \; else \; e_3 \Downarrow C}$$

$$\frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 = e_2 \Downarrow true} C_1 = C_2 \qquad \frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 = e_2 \Downarrow false} C_1 \neq C_2$$

$$\frac{e \Downarrow C}{isempty \; e \Downarrow true} C = \{\} \qquad \frac{e \Downarrow C}{isempty \; e \Downarrow false} C \neq \{\}$$

$$\frac{e \Downarrow \{C_1, \ldots, C_n\}}{powerset \; e \Downarrow \{C'_1, \ldots, C'_{2^n}\}}$$

where $C'_1, \ldots, C'_{2^n}$ are the subsets of $\{C_1, \ldots, C_n\}$

Recursive queries are costly in NRC(powerset)

- Theorem 13 (Suciu, Paredaens, PODS94)

  Any implementation of transitive closure in NRC(powerset) must use exponential space

- Theorem 14 (Van den Bussche, TCS01)

  Every flat relational query on unary schemas in NRC(powerset) is either already expressible in NRC w/o using the powerset operation or must use exponential space

- Theorem 15 (Biskup, Paredaens, Schwentick, Van den Bussche, SIAM J Comput 04)

  Any implementation of set parity in the "Equation Algebra" must use exponential space

- **These intensional expressive power results are quite query specific, and their proofs are not easily “portable” to other queries**

Motifs and Bounded Structures

- Given a signature τ. A "motif" of radius r is a first-order formula $\rho(u)$ with a single free variable u and has locality index r on all τ structures
- A τ structure $\mathcal{A}$ is "bounded" by a motif $\rho(u)$ at a threshold g if there are atmost rg elements in the universe of $\mathcal{A}$ that make $\rho(u)$ true, where r is the radius of $\rho(u)$
- A class $\mathcal{C}$ of τ structures is "bounded" by a motif $\rho(u)$ at a threshold g if $\rho(u)$ bounds all structures in $\mathcal{C}$ at the threshold g. On the other hand, $\mathcal{C}$ is said to be "unbounded" by $\rho(u)$ if for each $g > 0$, there is $\mathcal{A} \in \mathcal{C}$ that is not bounded by $\rho(u)$ at threshold g

Dichotomous Structures

- A class $\mathcal{C}$ of τ structures is "dichotomous" at threshold g iff
  (i) $\mathcal{C}$ is unbounded at threshold g by some motifs, and
  (ii) $\mathcal{C}$ is bounded by all other motifs at threshold g
- A dichotomous class $\mathcal{C}$ is "deep" if it is unbounded by some motifs of radius r at every r
- A dichomotomous class $\mathcal{C}$ is "severe" if for every motif $\rho(u)$ that unbounds $\mathcal{C}$, there is a sequence of structures $\mathcal{A}_1$, $\mathcal{A}_2$, ..., in $\mathcal{C}$ having universe of increasing size, and the ratio $|\{a \in \mathcal{A}_i \mid \mathcal{A}_i, [a/u] \models \rho(u)\}|/|A_i|$ tends to 1 as i tends to infinity.

- Deep severely dichotomous structures include long chains, long circles, deep trees, etc. Non-deep severely dichotomous structures include large sets of points, fat trees, etc.

Dichotomy Theorem

- Theorem 16 (Wong, PODS13)

  Let *f* be a flat relational query in NRC(Q,+,•,–,÷,Σ,=, $\leq^Q$, powerset) on structures from a class $\mathcal{C}$ where (i) $\mathcal{C}$ is severely dichotomous and (ii) its structures have degree ≤ k. Then either *f* is already expressible in NRC(Q,+,•,–,÷,Σ,=, $\leq^Q$) or must use exponential space

- Corollary 17

  All implementations of transitive closure, set parity, etc. in NRC(Q,+,•,–,÷,Σ,=, $\leq^Q$, powerset) must use exponential space

This theorem generalizes earlier intensional expressive power results

- Works for all queries on “severely dichotomous” structures
- Works for a more powerful query language
- Uses a proof technique that is “portable”

Another form of structural recursion mentioned in our ICDT92 paper

$$\frac{i : s \times t \to t \quad e : t}{sri(i, e) : \{s\} \to t}$$

- **Semantics**

$$\begin{aligned} sri(i,e)(\{\}) &= e \\ sri(i,e)(\{o\} \cup O) &= i(o, sri(i,e)(O)) \end{aligned}$$

- **In short…**

$$sri(i,e)\{o_1, \ldots, o_n\} = i(o_1, \ldots, i(o_n, e) \ldots)$$

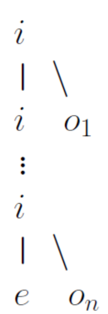
Equivalence

- Proposition 18 (Suciu, Wong, ICDT95)

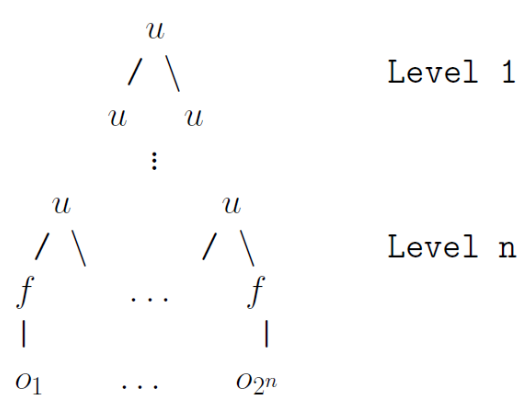
  There are uniform translations between NRC(sru) and NRC(sri). So for any set of external functions *Σ*, we have NRC(sru, Σ) = NRC(sri, Σ)

- **Our uniform sri → sru translation is expensive**

$sri(i, e)\{o_1, \ldots, o_n\}$ evaluates like this ...



$sru(u, f, e)\{o_1, \ldots, o_{2^n}\}$ evaluates like this ...



$sru(u, f, e)\{o_1, \ldots, o_{2^n}\}$ evaluates like this ...

Some sri queries cannot be parallelized

- Theorem 19 (Suciu, Wong, ICDT95)

  Any uniform translation of NRC(sri) queries to NRC(sru) / NRC(hom) must map some PTIME queries into EXPSPACE ones

- **In fact, in the presence of certain external functions, there is a PTIME NRC(sri) query for which every equivalent NRC(sru) / NRC(hom) query requires EXPSPACE**

NC is strictly in PTIME?

- Theorem 20 (Tannen, Suciu, PODS94)
  - **$NRC^1(hom, \leq)$ captures NC**
  - **$NRC^1(sri, \leq)$ captures PTIME**

- Corollary 21 (Suciu, Wong, ICDT95)

  There is no uniform translation of a language for PTIME into a language for NC

Notations: NRC^1 = the flat-types fragment of NRC

A cute result on lists

- **Treat *{}* as empty list, *{e}* as singleton list, $\cup$ as list concatenation. Then NRC(sru) and NRC(sri) become query languages for list**

- Theorem 20

  The *zip : {b}* $\times$ *{b}* → *{b* $\times$ *b}* function cannot be implemented in NRC(sru) and NRC(sri) in O(min(m, n)) time, where (m, n) are length of the two input lists to *zip*

Proof Idea

- **Suppose *zip* can be implemented in O(min(m,n)) time. Then *head : {b} → {b}* can be implemented in constant time in NRC(sri)**

$$\textit{head}\ (L) = \textit{sri}\ (\lambda x.\{\pi_1\ x\}, \{\})\ (\textit{zip}\ (L, \{\{\}\}))$$

- **But it is easy to show that *head* cannot be implemented in NRC(sri) in constant time**

ADDING ANNOTATIONS

What are annotations

- **Data can be annotated for many reasons**
  - Confidentiality policy
    - **Public < Confidential < Secret < Top Secret < 0**
  - Provenance
  - Probability
  - Uncertainty

- **It is desirable to propagate annotations on source tuples to query results**

Example

Source and Answer as K-Relations:

R

A	B	C	
a	b	c	x_1
d	b	e	x_2
f	g	e	x_3

S

B	C	
b	c	x_4
g	c	x_5

Q

A	C	
a	c	$x_1^2 + x_1 \cdot x_4$
a	e	$x_1 \cdot x_2$
d	c	$x_1 \cdot x_2 + x_2 \cdot x_4$
d	e	x_2^2
f	c	$x_3 \cdot x_5$
f	e	x_3^2

$$Q = \pi_{AC}(\pi_{AB}(R) \bowtie (\pi_{BC}(R) \cup S))$$

- "Thesis" 21 (Green, Karvounarakis, Tannen, PODS07)

  The propagation of a rich variety of annotations can be expressed as a semi-ring ⟨K, +, *, 0, 1⟩

How to propagate annotations for positive NRC

$$[\![l]\!]_K^\rho = l \qquad [\![x]\!]_K^\rho = \rho(x) \qquad [\![\{\}]\!]_K^\rho(x) = 0_K$$

$$[\![\{e\}]\!]_K^\rho(x) = \textit{if } x = [\![e]\!]_K^\rho \textit{ then } 1_K \textit{ else } 0_K$$

$$[\![e_1 \cup e_2]\!]_K^\rho(x) = [\![e_1]\!]_K^\rho(x) + [\![e_2]\!]_K^\rho(x)$$

$$\frac{[\![e_1]\!]_K^\rho = s_1}{[\![\bigcup(x \in e_1)\, e_2]\!]_K^\rho(y) = \sum_{v \in \mathsf{dom}(s_1)} s_1(v) \cdot [\![e_2]\!]_K^{\rho[x \leftarrow v]}(y)}$$

$$\frac{\textit{if } [\![e_1]\!]_K^\rho = [\![e_2]\!]_K^\rho \textit{ then } [\![e_3]\!]_K^\rho \textit{ else } [\![e_4]\!]_K^\rho = v}{[\![\textit{if } e_1 = e_2 \textit{ then } e_3 \textit{ else } e_4]\!]_K^\rho = v}$$

$$\frac{[\![e_1]\!]_K^\rho = v_1 \qquad [\![e_2]\!]_K^\rho = v_2}{[\![(e_1, e_2)]\!]_K^\rho = (v_1, v_2)} \qquad \frac{[\![e]\!]_K^\rho = (v_1, v_2) \qquad i \in \{1, 2\}}{[\![\pi_i(e)]\!]_K^\rho = v_i}$$

- Theorem 22 (Foster, Green, Tannen, PODS08)

If h : K1 → K2 is a homomorphism of semi-rings then h(e(v)) = h(e)(h(v))

Finer Notions of Provenance

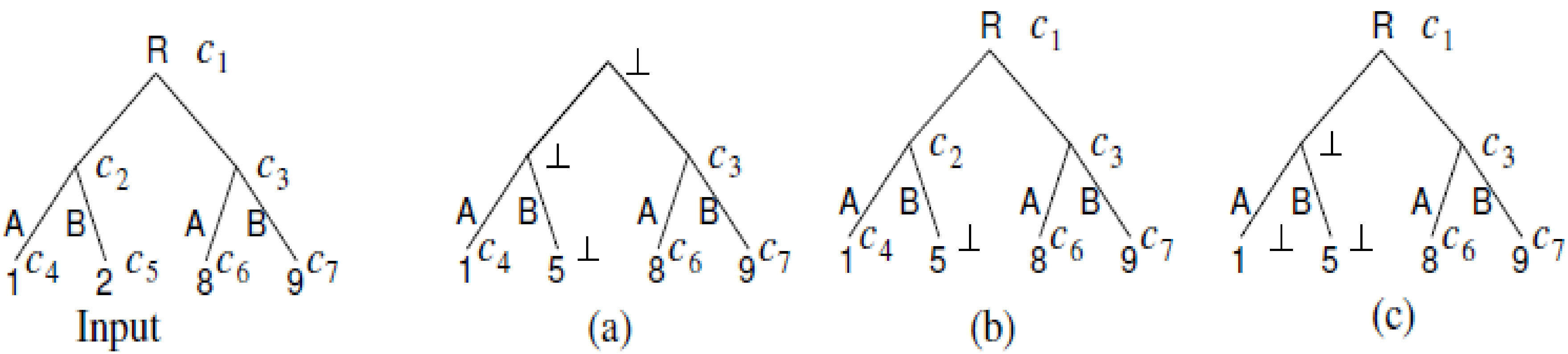
a) **(select * from R where A <> 1) union (select A, 5 as B from R where A = 1)**

b) **update R set B = 5 where A = 1**

c) **delete from R where A = 1; insert into R values (1, 5)**

Copying

Kind Preserving
If output item has same color as input item then they are of the same kind: both sets, both tuples, or identical atoms



Color propagation for query (a) and updates (b) and (c).

NRL(color) = NRC + NUL

- **Add these NUL constructs for updates to NRC**

$$\frac{}{\mathsf{skip}^s : s \to s} \qquad \frac{u_1 : r \to s \quad u_2 : s \to t}{u_1 ; u_2 : r \to t} \qquad \frac{e : t}{\mathsf{repl}^s\ e : s \to t} \qquad \frac{u : s \to t}{[x^s]\, u : s \to t}$$

$$\frac{e : \{s\}}{\mathsf{insert}\, e : \{s\} \to \{s\}} \qquad \frac{e : \{s\}}{\mathsf{remove}\, e : \{s\} \to \{s\}} \qquad \frac{u : s \to t}{\mathsf{iter}\ u : \{s\} \to \{t\}}$$

$$\frac{u : r \to t}{\mathsf{updl}^s\ u : r \times s \to t \times s} \qquad \frac{u : s \to t}{\mathsf{updr}^r\ u : r \times s \to r \times t}$$

- **Add a new type "color" to indicating provenance annotations**
  - $\perp$ is color to mean "newly created"
  - write $\underline{s}$ to mean type with provenance annotations

Provenance Semantics

$$\mathcal{P}[a] := (a, \bot) \qquad \mathcal{P}[\lambda x^s.e] := \lambda x^{\underline{s}}.\, \mathcal{P}[e]$$

$$\mathcal{P}[x^s] := x^{\underline{s}} \qquad \mathcal{P}[e_1\, e_2] := \mathcal{P}[e_1]\, \mathcal{P}[e_2]$$

$$\mathcal{P}[()] := ((), \bot) \qquad \mathcal{P}[\pi_1\, e] := \pi_1\, \pi_1\, \mathcal{P}[e]$$

$$\mathcal{P}[\pi_2\, e] := \pi_2\, \pi_1\, \mathcal{P}[e] \qquad \mathcal{P}[(e_1, e_2)] := \big((\mathcal{P}[e_1], \mathcal{P}[e_2]), \bot\big)$$

$$\mathcal{P}[\{\}] := (\{\}, \bot) \qquad \mathcal{P}[e_1 \cup e_2] := \big((\pi_1\, \mathcal{P}[e_1] \cup \pi_1\, \mathcal{P}[e_2]), \bot\big)$$

$$\mathcal{P}[\{e\}] := (\{\mathcal{P}[e]\}, \bot) \qquad \mathcal{P}[\bigcup\{e_2 \mid x^s \in e_1\}] := \big(\bigcup\{\pi_1\, \mathcal{P}[e_2] \mid x^{\underline{s}} \in \pi_1 \mathcal{P}[e_1]\}, \bot\big)$$

$$\mathcal{P}[\text{if } e_1 = e_2 \text{ then } e_3 \text{ else } e_4] := \text{if } val(\mathcal{P}[e_1]) = val(\mathcal{P}[e_2]) \text{ then } \mathcal{P}[e_3] \text{ else } \mathcal{P}[e_4]$$

$$\mathcal{P}[\mathsf{skip}^s] := \mathsf{skip}^{\underline{s}} \qquad \mathcal{P}[u; u'] := \mathcal{P}[u]; \mathcal{P}[u']$$

$$\mathcal{P}[\mathsf{repl}^s\ e] := \mathsf{repl}^{\underline{s}}\ \mathcal{P}[e] \qquad \mathcal{P}[[x^s]\, u] := [x^{\underline{s}}]\, \mathcal{P}[u]$$

$$\mathcal{P}[\mathsf{insert}\ e] := \mathsf{updl}\ \mathsf{insert}\, (\pi_1\, \mathcal{P}[e]) \qquad \mathcal{P}[\mathsf{iter}\ u] := \mathsf{updl}\ \mathsf{iter}\ \mathcal{P}[u]$$

$$\mathcal{P}[\mathsf{updl}\ u] := \mathsf{updl}\ \mathsf{updl}\ \mathcal{P}[u] \qquad \mathcal{P}[\mathsf{updr}\ u] := \mathsf{updl}\ \mathsf{updr}\ \mathcal{P}[u]$$

$$\mathcal{P}[\mathsf{remove}\ e] := \mathsf{updl}\ \big([x]\ \mathsf{remove}\, \{y \mid y \in x, val(y) \in val(\mathcal{P}[e])\}\big)$$

Provenance-Aware DB Operations

- ***f*** **:** $\underline{s}$ **→** $\underline{t}$ **is color propagating if *f* does not let input colors influence the uncolored part of the output and *f* is insensitive to actual colors used**
- ***f*** **:** $\underline{s}$ **→** $\underline{t}$ **is bounded inventing if *f* does not create many new values**
- **A provenance-aware db operation (pado) is a color-propagating and bounded-inventing function *f* :** $\underline{s}$ **→** $\underline{t}$

Soundness and Completeness

- Theorem 23 (Buneman, Cheney, VanSummeren, ICDT07)

  Every function is in CP if and only if it is in PNRC

- Theorem 24 (Buneman, Cheney, VanSummeren, ICDT07)

  Every function is in KP if and only if it is in PNUL

$$\mathcal{CP} := \{\ f \mid f : \underline{s} \to \underline{t} \text{ in } \mathcal{NRL}(color) \text{ defines a copying pado }\},$$
$$\mathcal{KP} := \{\ f \mid f : \underline{s} \to \underline{t} \text{ in } \mathcal{NRL}(color) \text{ defines a kind-preserving pado }\}.$$

$$\mathcal{PNRC} := \{\ \mathcal{P}[e] \mid e \text{ expression in } \mathcal{NRC}\ \}.$$
$$\mathcal{PNUL} := \{\mathcal{P}[u] \mid u \text{ expression in } \mathcal{NUL}\}.$$

OPEN PROBLEMS

Maybe you know the answer …

- **In the presence of an order on base types,**
  - **Locality theorem becomes useless**
  - **Bounded degree property fails**
  - **Dichotomy theorem fails**

  Can this be fixed?

- **Is there a PTIME query in NRC(sri) that has no PTIME equivalent in NRC(sru) in the absence of external functions? Is transitive closure such a query?**