A Query Language for NC

Dan Suciu Val Breazu-Tannen
University of Pennsylvania

Motivation

- FO and equivalent formalisms accepted as theoretical core for query languages since '70.
- But FO not expressive enough: transitive closure, parity, etc., are not expressible in FO.
- Add iterative constructs:
  - Fixpoints [Immerman86, Vardi82]:

$$FO + LFP + \leq \; = \; PTIME$$
$$FO + PFP + \leq \; = \; PSPACE$$

$\Rightarrow$ fixpoints fit **sequential** query evaluation.

Motivation (cont'd)

- But what fits **parallel** evaluation ? Here we propose: **divide and conquer recursion on sets**.
- This is an old construct. Called:
  - *pump* in FAD
    [Bancilhon, Briggs, Khoshafian, Valduriez]
  - *hom* in MACHIAVELLI
    [Ohori, Buneman, Breazu-Tannen]
  - a form of *transducer* in SVP
    [Parker, Simon, Valduriez]

Two Examples of Divide and Conquer Recursion on Sets

Parity

$$\begin{cases} parity(\emptyset) & \stackrel{\text{def}}{=} \; false \\ parity(\{x\}) & \stackrel{\text{def}}{=} \; true \\ parity(S_1 \cup S_2) & \stackrel{\text{def}}{=} \; parity(S_1) \; xor \; parity(S_2) \\ & \quad \text{when } S_1 \cap S_2 = \emptyset \end{cases}$$

E.g. $parity(\{a, b, c, d, e\})$ can be computed as follows:

Note that there are several ways a set S can be decomposed during a divide and conquer recursion.

Two Examples of Divide and Conquer Recursion on Sets (cont'd)

Transitive closure

Given a binary relation R, compute its transitive closure $tc(R)$ by:

$$tc(R) \stackrel{\text{def}}{=} \varphi(V)$$

where V and φ are defined by:

$$V \stackrel{\text{def}}{=} \Pi_1(R) \cup \Pi_2(R)$$

$$\begin{cases} \varphi(\emptyset) & \stackrel{\text{def}}{=} \emptyset \\ \varphi(\{x\}) & \stackrel{\text{def}}{=} R \\ \varphi(S_1 \cup S_2) & \stackrel{\text{def}}{=} \varphi(S_1) \cup \varphi(S_2) \cup \varphi(S_1) \circ \varphi(S_2) \\ & \quad \text{when } S_1 \cap S_2 = \emptyset \end{cases}$$

Note that, when $card(S) = k$, then

$$\varphi(S) = R \cup R^2 \cup \ldots \cup R^k$$

Formal definition of Divide and Conquer Recursions

Given e, f, u, define $\varphi \stackrel{\text{def}}{=} dcr(e, f, u)$

$$\begin{cases} \varphi(\emptyset) & \stackrel{\text{def}}{=} & e \\ \varphi(\{x\}) & \stackrel{\text{def}}{=} & f(x) \\ \varphi(S_1 \cup S_2) & \stackrel{\text{def}}{=} & u(\varphi(S_1), \varphi(S_2)) \\ & & \text{when } S_1 \cap S_2 = \emptyset \end{cases}$$

Need to check :

$$\begin{array}{lclcl} u(a, u(b, c)) & = & u(u(a, b), c) & & \text{(associativity)} \\ u(a, b) & = & u(b, a) & & \text{(commutativity)} \\ u(e, a) & = & u(a, e) & = a & \text{(identity)} \end{array}$$

Checking the conditions is undecidable, not even r.e. !

The Rest of the Language

- desired: same expressive power as FO, at flat types.
- but in a formalism and type system compatible with the presentation of dcr [Breazu-Tannen, Buneman, Wong 92].

The Types

$$
\begin{array}{rcll}
t & ::= & D & \text{the base type} \\
 & | & bool & \text{booleans} \\
 & | & t \times t & \text{product type} \\
 & | & \{t\} & \text{set type}
\end{array}
$$

Restrictions of the set height

Set height ≤ 1 correspond to flat relations and scalar values. E.g. $t = D \times \{D \times D\}$.

Arbitrary set heights, correspond to **complex objects**. E.g. $t = \{D \times \{D\}\}$.

The Rest of the Language (cont'd)

The Language $\mathcal{NRA}$

$$\frac{}{x^t : t} \qquad \frac{e_1 : t_1 \quad e_2 : t_2}{(e_1, e_2) : (t_1, t_2)}$$

$$\frac{e : t_1 \times t_2}{\pi_1(e) : t_1} \qquad \frac{e : t_1 \times t_2}{\pi_2(e) : t_2}$$

$$\frac{}{\emptyset : \{t\}} \qquad \frac{e : t}{\{e\} : \{t\}} \qquad \boxed{\frac{e_1 : \{t\} \quad e_2 : \{t\}}{e_1 \cup e_2 : \{t\}}}$$

$$\frac{e_1 : D \quad e_2 : D}{e_1 = e_2 : bool} \qquad \frac{}{() : unit}$$

$$\frac{e : \{t\}}{empty(e) : bool} \qquad \frac{e : bool \quad e_1 : t \quad e_2 : t}{if\ e\ then\ e_1\ else\ e_2 : t}$$

$$\frac{e : t}{\lambda x^s.e : s \to t} \qquad \frac{f : s \to t \quad e : s}{f(e) : t}$$

$$\boxed{\frac{f : s \to \{t\}}{ext(f) : \{s\} \to \{t\}}}$$

Explanations

- $\cup$ is *union*
- $ext(f)(\{x_1, \ldots, x_n\}) \stackrel{\text{def}}{=} f(x_1) \cup \ldots \cup f(x_n)$

The Rest of the Language (cont'd)

Connection with other languages

- At **flat types**: $\mathcal{NRA}^1$ has the same expressive power as FO.

$$\boxed{\mathcal{NRA}^1 = FO}$$

- At all types (**complex object types**): $\mathcal{NRA}$ has the same expressive power as other formalisms for tractable language for complex objects (Abiteboul and Beeri's **algebra without powerset**, Scheck and Scholl's NF^2, Thomas and Fischer's *algebra*, Paredaens and Van Gucht's *nested algebra*).

The language for NC at flat types: $\mathcal{NRA}^1(dcr)$

Review: The classes NC, AC^k

Definition 1

- Let $k \geq 0$. Then $AC^k =$ the functions computable on a CRCW PRAM:
  - in time $O(\log^k n)$, and
  - with polynomially many processors

  [Stockmeyer and Vishkin 84]
- $NC = \cup_{k \geq 0} AC^k$.

Comments

- $AC^0 \subset AC^1 \subseteq \ldots \subseteq NC \subseteq PTIME$
- Just as PTIME is considered as the class of sequential tractable functions, NC is considered the class of parallel tractable functions.

Main Result

Theorem 1

- $\mathcal{NRA}^1(dcr, \leq) = NC$
- $\mathcal{NRA}^1(dcr^{(k)}, \leq) = AC^k,\ \forall k \geq 1$

(where $dcr^{(k)}$ means that dcr may be nested at most k levels).

In contrast:

Proposition 2 (This is related to [Immemrman, Patnaik, Stemple 91])

- $\mathcal{NRA}^1(sri, \leq) = \mathcal{NRA}^1(sri^{(1)}, \leq) = PTIME$

where, for given e, i, the structural recursion on the insert presentation $\varphi = sri(e, i)$ is defined by:

$$\begin{cases} \varphi(\emptyset) & \stackrel{\text{def}}{=} \quad e \\ \varphi(\{x\} \cup S) & \stackrel{\text{def}}{=} \quad i(x, \varphi(S)) \end{cases}$$

(Need to check conditions similar to those for sru.)

NC v.s. $PTIME$

- The difference between NC and $PTIME$ reduces to the difference between two ways of recurring over sets.
- [Breazu-Tannen & Subrahmanian 91] give a translation of dcr into sri, implying that $\mathcal{NRA}^1(dcr) \subseteq \mathcal{NRA}^1(sri)$. But it is unlikely that dcr can simulate efficiently sri, because it is unlikely that $NC = PTIME$.

Comments on the Main Result

- Without order, $\mathcal{NRA}^1(dcr) \subseteq NC$. This holds even in the presence of external functions in NC, e.g. $\mathcal{NRA}^1(int, +, -, *, div; dcr) \subseteq NC$. But we need *order* to capture the whole class NC with dcr [Immerman, Patnaik, Stemple 91].
- dcr allows iteration of a function $\log n$ times, i.e. $g(x) \stackrel{\text{def}}{=} f^{(\log n)}(x)$, for $n = card(S)$.
- We express $f^{(\log^2 n)}(x)$ with two levels of dcr nesting.
- sri allows iteration of a function n times, i.e. $g(x) \stackrel{\text{def}}{=} f^{(n)}(x)$, for $n = card(S)$.
- Can get $f^{(n^2)}(x)$ still with one nesting level by taking $S \times S$.

dcr, sri can both compute $powerset$ over complex objects. To capture NC and $PTIME$, we define "bounded" versions of dcr and sri, which are tractable.

Given e, f, u and a set B, define φ by **bounded divide and conquer recursion** [Buneman]:

$$\begin{cases} \varphi(\emptyset) & \stackrel{\text{def}}{=} \quad e \cap B \\ \varphi(\{x\}) & \stackrel{\text{def}}{=} \quad f(x) \cap B \\ \varphi(S_1 \cup S_2) & \stackrel{\text{def}}{=} \quad u(\varphi(S_1), \varphi(S_2)) \cap B \\ & \qquad \text{when } S_1 \cap S_2 = \emptyset \end{cases}$$

Need to check conditions.

Similar, define **bounded structural recursion on the insert presentation**.

Theorem 3

- $\mathcal{NRA}(bdcr, \leq) = NC$
- $\mathcal{NRA}(bsri, \leq) = PTIME$

Conclusions and open problems

- $\mathcal{NRA}^1(dcr)$ and $\mathcal{NRA}(bdcr)$ are not r.e, hence the results here are not so nice as e.g. $(FO + LFP + \leq) = PTIME$.

  However, there are r.e. subsets of $\mathcal{NRA}^1(dcr)$ and $\mathcal{NRA}(bdcr)$ which capture NC over ordered databases.
- Difficult open problem: is there a r.e. language $\mathcal{L}$ expressing exactly the NC queries over all databases ?
- Recall [Abiteboul and Vianu 91]:

$$FO + LFP \neq FO + PFP \iff PTIME \neq PSPACE$$

  Open problem:

$$\mathcal{NRA}^1(dcr) \neq \mathcal{NRA}^1(sri) \overset{?}{\iff} NC \neq PTIME$$