Any Algorithm in the Complex Object Algebra with Powerset Needs Exponential Space to Compute Transitive Closure

MS-CIS-94-04
LOGIC & COMPUTATION 76

Dan Suciu
Jan Paredaens

University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

February 1994

Any Algorithm in the Complex Object Algebra with Powerset Needs Exponential Space to Compute Transitive Closure

Dan Suciu*
Dept. of Comp. and Info. Science
University of Pennsylvania, USA
suciu@saul.cis.upenn.edu

Jan Paredaens
Dept. of Computer Science
University of Antwerp, Belgium
pareda@wins.uia.ac.be

December 1, 1993

Abstract

The Abiteboul and Beeri algebra for complex objects can express a query whose meaning is transitive closure, but the algorithm naturally associated to this query needs exponential space. We show that any other query in the algebra which expresses transitive closure needs exponential space. This proves that in general the powerset is an intractable operator for implementing fixpoint queries.

1 Introduction

Abiteboul and Beeri in [1] have shown that powerset can express transitive closure (tc), in a language for complex objects without fixpoints or any other form of iterations. But the obvious way of doing that is by a query whose naturally associated algorithm requires exponential space (and time). We prove here that in order to express tc with powerset, exponential space (and time) is indeed needed.

This result is of a different nature than classical inexpressibility results (like transitive closure is not expressible in FO ([3]) or *even* is not expressible in FO+LFP, because it says that transitive closure, although expressible in a particular language, is not expressible *efficiently* in that language. This denotes a mismatch between the complexity of the natural way of computing queries in that language, and the complexity of the best Turing Machine for computing those queries. It is in the same spirit as Abiteboul and Vianu's result that *even* cannot be computed in polynomial space on a generic machine ([2]).

Technically, our result is slightly stronger, in that it proves that powerset cannot express efficiently *deterministic* transitive closure (see [8]), i.e. transitive closure of a graph whose nodes have outdegree ≤ 1. More precisely, we prove that in order to compute the transitive closure of the relation $r_n = \{(0,1),(1,2),\ldots,(n-1,n)\}$ in a certain language with powerset, exponential space is needed. In addition, in any query over r_n which does not require exponential space, we can replace all occurrences of powerset with some approximation expressible in that language without powerset.

*The author was partially supported by grant NSF CCR-90-57570

A consequence of our results is that the powerset is not an efficient operator for the implementation of fixpoint queries in general. Clearly, adding *while* to the algebra, instead of *powerset*, gives us the same computational power but it evidently only uses polynomial time (and space) for computing transitive closure.

We conjecture that *any* query expressible efficiently with powerset is expressible also without powerset. However, this problem remains open.

In sections 2 and 3 we define the *nested relational algebra with powerset* (which has the same expressive power as Abiteboul and Beeri's *algebra*) and the complexity of its evaluation. We state our main results in section 4, and prove them in section 5. The proof follows from three facts: (1) the *abstract expressions*, defined in section 5.1 are closed under application of functions in the nested relational algebra (this is shown in subsection 5.2), (2) the relation r_n can be expressed as an abstract expression, but its transitive closure cannot (this is shown in subsection 5.3), and finally (3) any function in the nested relational algebra with powerset applied to some abstract expression either yields another abstract expression, or has exponential complexity (this is shown in subsection 5.4).

2 The Language

Abiteboul and Beeri, in [1], define the *complex values algebra*, as a functional language for complex objects, and show that it has the same expressive power as the *domain independent calculus*, an extension of first order logic to complex objects. Powerset is explicitly included in the algebra, and the authors show how powerset can be used to compute transitive closure. The nested relational algebra $\mathcal{NRA}$ which we consider here has the same expressive power as the *algebra without powerset* in [1], while $\mathcal{NRA}(powerset)$ has the same expressive power as the *algebra*. $\mathcal{NRA}$ has essentially the same expressive power as Schek and Scholl's NF^2 relational algebra ([13]), as Thomas and Fischer's algebra ([14]), and as Paredaens and Van Gucht's *nested algebra* ([11], [12]). In defining $\mathcal{NRA}$, we follow the formalism in [6].

The **nested relational algebra** $\mathcal{NRA}$ is a typed language. Its types are build from the following **base types**: $\mathbb{B}$ (the booleans), *unit* (the single-valued type, $unit = \{()\}$), and $\mathbb{N}$ (the natural numbers), and are given then by the grammar:

$$t ::= unit \mid \mathbb{B} \mid \mathbb{N} \mid t \times t \mid \{t\}$$

The values of the type $s \times t$ are pairs (x, y), with $x \in s$ and $y \in t$, while the values of the type $\{t\}$ are finite sets of elements from t.

$\mathcal{NRA}$ is a variable-free language, and its expressions are functions $f : s \to t$. It contains some primitive functions (like union $\cup^s : \{s\} \times \{s\} \to \{s\}$ for all types s), as well as formation rules, like the rule saying that the composition of two functions, $g \circ f : r \to t$, is in the language, whenever both $f : r \to s$ and $g : s \to t$ are in the language. Note that functions of the type $unit \to s$ correspond to values of type s. $\mathcal{NRA}$ is defined below:

$$\frac{}{id^s : s \to s} \qquad \frac{f : r \to s \quad g : s \to t}{g \circ f : r \to t}$$

$$\frac{}{!^s : s \to unit} \qquad \frac{f : r \to s \quad g : r \to t}{\langle f, g \rangle : r \to s \times t} \qquad \frac{}{\pi_1^{s,t} : s \times t \to s} \qquad \frac{}{\pi_2^{s,t} : s \times t \to t}$$

$$\frac{f : s \to t}{map(f) : \{s\} \to \{t\}} \qquad \frac{}{\eta^s : s \to \{s\}} \qquad \frac{}{\mu^s : \{\{s\}\} \to \{s\}} \qquad \frac{}{\rho_2^{s,t} : s \times \{t\} \to \{s \times t\}}$$

$$\frac{}{\emptyset^s : unit \to \{s\}} \quad \frac{}{\cup^s : \{s\} \times \{s\} \to \{s\}} \quad \frac{}{=: \mathbb{N} \times \mathbb{N} \to \mathbb{B}} \quad \frac{}{empty : \{s\} \to \mathbb{B}}$$

$$\frac{}{true : unit \to \mathbb{B}} \quad \frac{}{false : unit \to \mathbb{B}} \quad \frac{f : s \to \mathbb{B} \quad f_1 : s \to t \quad f_2 : s \to t}{if\ f\ then\ f_1\ else\ f_2 : s \to t}$$

We briefly describe the meaning of these functions. *id* is the identity function, ! is the constant function such that $!(x) = ()$. $\langle f, g\rangle$ is the pair formation function such that $\langle f, g\rangle(x) = (f(x),\ g(x))$. π_1 and π_2 are respectively first and second projection. $map(f)\{x_1, \cdots, x_n\}$ is defined to be $\{f(x_1), \cdots, f(x_n)\}$: it is called *replace* ($\rho\langle f\rangle$) in [1]. η is the singleton formation function such that $\eta(x) = \{x\}$. $\mu\{x_1, \cdots, x_n\} = x_1 \cup \ldots \cup x_n$ flattens a set of sets: it is called *set-collapse* in [1]. Further, $\rho_2(x, \{y_1, \cdots, y_n\})$ is $\{(x, y_1), \ldots, (x, y_n)\}$, $\emptyset$ is the empty set, $\cup$ is set union, $= (x, y)$ returns *true* iff $x = y$ and *false* otherwise, $empty(x)$ returns true iff $x = \emptyset$, *true*, and *false* are constants. Finally, *if f then f_1 else f_2* is the function g such that, $\forall x$, if $f(x)$ is true then $g(x) = f_1(x)$ and otherwise $g(x) = f_2(x)$.

This language has the expressive power of the algebra without powerset of [1]:

Proposition 2.1 *The following operations are definable in $\mathcal{NRA}$: the database projections, cartesian product, equality at all types, set difference, set intersection, set membership, set inclusion, selection over any predicate definable in $\mathcal{NRA}$, nest, unnest. See [6].*

Now we consider a new primitive operation, the powerset:

$$\frac{}{powerset : \{s\} \to \{\{s\}\}}$$

and denote with $\mathcal{NRA}$(*powerset*) the language $\mathcal{NRA}$ extended with *powerset*. While all queries expressible in $\mathcal{NRA}$ are in PTIME, $\mathcal{NRA}$(*powerset*) can obviously express exponential queries. More interestingly, $\mathcal{NRA}$(*powerset*) can express PTIME queries, which are not expressible in $\mathcal{NRA}$: Abiteboul and Beeri prove in [1] that transitive closure can be expressed in $\mathcal{NRA}$(*powerset*). In contrast, we know from Paredaens [12] and Wong [15], that transitive closure is not expressible in $\mathcal{NRA}$. But the obvious way of expressing transitive closure in $\mathcal{NRA}$(*powerset*) is through an exponential space query. We prove in the following that exponential space is indeed needed.

3 The Complexity of Evaluation in $\mathcal{NRA}$(*powerset*)

In order to define the complexity of an evaluation in $\mathcal{NRA}$(*powerset*), we present an eager evaluation strategy for this language, and a complexity measure on the complex objects. Thus, our main result will depend (1) on the particular evaluation strategy and (2) on the complexity measure. "Reasonable" complexity measures for complex objects are polynomially related, so our result still holds for other reasonable complexity measures. In contrast, it is not obvious whether it still holds for a lazy evaluation strategy.

We define a denotation C for a **complex object** by the grammar:

$$C ::= x \mid false \mid true \mid () \mid (C, C) \mid \{C, \ldots, C\}$$

where $x \in \mathbb{N}$. We consider only well-typed complex objects. No duplicates are allowed in the denotation of complex objects of set type, i.e. in $\{C_1, \ldots, C_n\}$, C_i and C_j must denote distinct objects, when $i \neq j$. The **size** of some complex object C is defined by: $size(x) = size(false) = size(true) = size(()) = 1$, $size((C_1, C_2)) = 1 + size(C_1) + size(C_2)$ and $size(\{C_1, \ldots, C_n\}) = 1 + size(C_1) + \ldots + size(C_n)$.

We consider to be equal those complex objects denoted by sets with different orders of their elements, e.g. $\{C_1, C_2\}$ and $\{C_2, C_1\}$. Note that this does not affect the definition of the size.

In defining the eager evaluation strategy of $\mathcal{NRA}(powerset)$ we adopt the natural semantics style, as found in [10]. Under this style, for some function $f \in \mathcal{NRA}(powerset)$ and complex objects C, C', we write $f(C) \Downarrow C'$ to mean "$f(C)$ evaluates to C'". The binary relation $\Downarrow$ is defined by the following set of rules:

$$\frac{}{id(C) \Downarrow C} \qquad \frac{f(C) \Downarrow C' \quad g(C') \Downarrow C''}{(g \circ f)(C) \Downarrow C''}$$

$$\frac{}{!\,C \Downarrow ()} \qquad \frac{f_1(C) \Downarrow C_1 \quad f_2(C) \Downarrow C_2}{\langle f_1, f_2 \rangle(C) \Downarrow (C_1, C_2)} \qquad \frac{}{\pi_1(C_1, C_2) \Downarrow C_1} \qquad \frac{}{\pi_2(C_1, C_2) \Downarrow C_2}$$

$$\frac{f(C_1) \Downarrow C_1' \cdots f(C_n) \Downarrow C_n'}{map(f)\{C_1, \cdots, C_n\} \Downarrow \{C_1'\} \cup \cdots \cup \{C_n'\}}$$

$$\frac{}{\eta(C) \Downarrow \{C\}} \qquad \frac{}{\mu\{C_1, \ldots, C_n\} \Downarrow C_1 \cup \cdots \cup C_n} \qquad \frac{}{\rho_2(C, \{C_1, \cdots, C_n\}) \Downarrow \{(C, C_1), \cdots, (C, C_n)\}}$$

$$\frac{}{\emptyset() \Downarrow \{\}} \qquad \frac{}{\cup(C_1, C_2) \Downarrow C_1 \cup C_2} \qquad \frac{}{=(x, y) \Downarrow true} \text{ where } x = y. \qquad \frac{}{=(x, y) \Downarrow false} \text{ where } x \neq y.$$

$$\frac{}{empty(\{\}) \Downarrow true} \qquad \frac{}{empty(\{C_1, \ldots\}) \Downarrow false} \qquad \frac{}{true() \Downarrow true} \qquad \frac{}{false() \Downarrow false}$$

$$\frac{f(C) \Downarrow true \quad f_1(C) \Downarrow C'}{if\ f\ then\ f_1\ else\ f_2(C) \Downarrow C'} \qquad \frac{f(C) \Downarrow false \quad f_2(C) \Downarrow C'}{if\ f\ then\ f_1\ else\ f_2(C) \Downarrow C'}$$

$$\frac{}{powerset(\{C_1, \ldots, C_n\}) \Downarrow \{C_1', \ldots, C_{2^n}'\}} \text{ where } C_1', \ldots, C_{2^n}' \text{ are the subsets of } \{C_1, \ldots, C_n\}$$

Thus, an evaluation $f(C) \Downarrow C'$ (which we sometimes abbreviate $f(C) \Downarrow$), can be viewed as a tree, whose nodes are labeled by the rules above, and whose root contains $f(C) \Downarrow C'$. The *height* of the tree depends only on f, not on C. But the width of this tree may depend on C: the branching factor at each node may depend on the size of the complex object(s) at that node (see the *map* rule).

The **complexity** of some evaluation $f(C) \Downarrow$ is defined to be the size of the largest complex object occurring in the derivation tree of $f(C) \Downarrow$. This complexity measure is robust: e.g. the total number of nodes of the evaluation tree is polynomially bounded by this complexity, while the sum of the sizes of all complex objects in a tree is polynomially related to it.

4 Main Results

Let $r_n \stackrel{\text{def}}{=} \{(0,1), (1,2), (2,3), \ldots, (n-1, n)\}$, i.e. r_n is a complex object of type $\{\mathbb{N} \times \mathbb{N}\}$ representing a particular binary relation (a chain). Also, let $q_n \stackrel{\text{def}}{=} tc(r_n)$ be its transitive closure, $q_n = \{(x, y) \mid 0 \leq x < y \leq n\}$. Our main result is:

Theorem 4.1 *For any function $f \in \mathcal{NRA}(powerset)$ of type $f : \{\mathbb{N} \times \mathbb{N}\} \to \{\mathbb{N} \times \mathbb{N}\}$ such that $f(r_n) \Downarrow q_n$ for every $n > 0$, the complexity of $f(r_n) \Downarrow$ is $\Omega(2^{cn})$, for some $c > 0$.*

The proof is given in section 5. As a consequence, transitive closure is not expressible in an efficient way in $\mathcal{NRA}(powerset)$. In fact, the theorem implies the stronger result that *deterministic* transitive closure (i.e. transitive closure of a graph whose nodes have outdegree ≤ 1, see [8]) is not efficiently expressible.

We prove an additional interesting property. For any number $m \geq 0$, define the mth approximation of $powerset : \{s\} \to \{\{s\}\}$ to be some $\mathcal{NRA}$ function $powerset_m$, which returns all subsets of cardinality $\leq m$. Formally, $powerset_0(x) \stackrel{\text{def}}{=} \{\emptyset\}$ and $powerset_{m+1}(x) \stackrel{\text{def}}{=} \{\{u\} \cup s \mid u \in x, s \in powerset_m(x)\}$. For some function $f \in \mathcal{NRA}(powerset)$, define the mth approximation of f, f_m, to be the result of replacing all occurrences of $powerset$ in f with $powerset_m$.

Proposition 4.2 *For any function $f : \{\mathbb{N} \times \mathbb{N}\} \to s$ in $\mathcal{NRA}(powerset)$, either there exists some approximation f_m of f such that $f_m(r_n) = f(r_n)$, $\forall n > 0$, or the complexity of $f(r_n) \Downarrow$ is $\Omega(2^{cn})$, for some $c > 0$.*

The proof is given in section 5. We also conjecture that all efficient functions in $\mathcal{NRA}(powerset)$ are already in $\mathcal{NRA}$, but it is not clear whether the techniques used in proving theorem 4.1 can be generalized.

Could separation results from complexity theory offer us a shorter proof of theorem 4.1 ? The question is motivated by the observation that one can use such separation results to prove that transitive closure is not expressible in $\mathcal{NRA}$, as follows. Recall that the class AC^0 is the class of functions $f : \{0,1\}^* \to \{0,1\}^*$ computable by a "uniform" family of circuits made of NOT gates and unbounded fan-in AND and OR gates, having polynomial size and constant depth (see [4]): it is easy to see that $AC^0 \subseteq NLOGSPACE$, where $NLOGSPACE$ is the class of functions computable by a nondeterministic Turing Machine with $O(\log n)$ work space (see, e.g. [7]). It is known that transitive closure is *complete* for $NLOGSPACE$, w.r.t. first order reductions ([8]), hence, if transitive closure were expressible in $\mathcal{NRA}$, then $\mathcal{NRA} = NLOGSPACE$. By generalizing Immerman's result that $FO \subseteq AC^0$ ([9], [4]), we can prove $\mathcal{NRA} \subseteq AC^0$. On the other hand, using some result by Furst, Saxe and Sipser, and independent by Ajitai, one can show that $AC^0 \neq NLOGSPACE$ (see [4]); hence transitive closure cannot be expressed in $\mathcal{NRA}$.

Trying to reason along the same lines for the tractable fragment of $\mathcal{NRA}(powerset)$, we consider the class TC^0, which is defined similarly to AC^0, but by allowing the circuits to contain an additional type of gates, the *threshold gates*: a threshold gate is labeled by some number k, and its output is 1 iff at least k of its inputs are 1 (see [4]). The following hold: $AC^0 \subset TC^0 \subseteq NLOGSPACE$. We can prove:

Proposition 4.3 *All functions in $\mathcal{NRA}(powerset)$ having polynomially bounded complexity are in TC^0.*

But this does not suffice to prove that transitive closure is not efficiently expressible in $\mathcal{NRA}(powerset)$, because it is still open whether $TC^0 \neq NLOGSPACE$. However, a proof of the fact that $TC^0 \neq NLOGSPACE$ would imply a weaker version of theorem 4.1, namely that transitive closure is not expressible in $\mathcal{NRA}(powerset)$ with polynomially bounded complexity.

5 Proof of the Main Theorem

Recall that $r_n = \{(0,1),(1,2),\ldots,(n-1,n)\}$, for all $n > 0$. The idea is to express, syntactically, all possible complex objects which can occur in the evaluation tree of $f(r_n)$, without using space exponential in n, for all $f \in \mathcal{NRA}(powerset)$, and to observe that $tc(r_n)$ is not among them. For this, we develop the language of *abstract* *expressions*. Think of an abstract expression A of type s as denoting some complex object of types s, for every $n > 0$. Examples of abstract expressions are $\{(2, x, y) \mid x = 0, n; y = 0, n\}$ and $\{(x, x+1) \text{ when } x \neq n \mid x = 0, n\}$. The latter denotes r_n, for each $n > 0$.

5.1 Abstract Expressions

We consider an infinite set of **variables** to be given, like $x, y, \alpha, \beta \ldots$, ranging over the set $[n] \overset{\text{def}}{=} \{0, 1, 2, \ldots, n\}$, and define **simple expressions** e to be (1) a positive number c or (2) $n - c$ where c is a positive number or (3) $x + c$ where x is a variable and c is a number. E.g. $7, n - 9, n, x, x + 3, y - 8$ are simple expressions. But $x + y, n - x, 2 * x$ are not.

We define a **simple condition** to be a condition of the form $e = e'$, or $e \neq e'$, where e, e' are simple expressions. A **condition** is obtained by combining simple conditions with $\vee$ (or), $\wedge$ (and), *true* and *false*. E.g. $x = y+5 \wedge y \neq z - 1 \vee x \neq y + 1 \wedge y = z + 5$ is a condition.

An **abstract expression** is: $()$, e (where e is a simple expression), *true*, *false*, (A_1, A_2) where A_1, A_2 are abstract expressions, $\{A \mid x_1 = 0, n; \ldots x_k = 0, n\}$ (when $k = 0$ this becomes the singleton set $\{A\}$), $A_1 \cup A_2$ and $(A_1 \text{ when } C_1; A_2 \text{ when } C_2; \ldots A_l \text{ when } C_l)$, where $A_1, \ldots, A_l$ are abstract expressions and $C_1, \ldots, C_l$ are pairwise disjunctive conditions (i.e. $C_i \wedge C_j$ is equivalent to *false*, for $i \neq j$). The latter construct is called **guarded expression**, and the conditions $C_1, \ldots C_l$ are called the **guards**.

Examples of abstract expressions are: 3, $n - 5$, $\{(x, x+2) \text{ when } (x \neq n \wedge x \neq n - 1) \mid x = 0, n\}$. Also $\{2, 5, 12\}$ is an abbreviation for the abstract expressions $\{2\} \cup \{5\} \cup \{12\}$. We only consider *typed* abstract expressions, e.g. $\{A \mid x = 0, n\}$ is an abstract expression of type $\{s\}$ provided that A is an abstract expression of type s. As usual, we distinguish **bound** and **free** variables in some abstract expression A.

Let $n > 0$, A be an abstract expression of type s and let ρ be an *environment*, i.e. some function assigning values in $[n]$ to the free variables of A. Then, we associate to each abstract expression A of type s, a complex object $[\![A]\!]\rho$ of type s, in the obvious way. $[\![A]\!]\rho$ may not be defined (e.g. when no guard in a guarded abstract expression is true). E.g., when $\rho(x) = 1$, then $[\![\{(x, y) \text{ when } x \neq y \mid y = 0, n\}]\!]\rho = \{(1,0), (1,2), (1,3), \ldots, (1,n)\}$. Clearly, when A is closed, then $[\![A]\!]\rho$ does not depend on ρ, and we abbreviate it with $[\![A]\!]$. E.g. $[\![\{0 \text{ when } \mathit{false}\}]\!] = \emptyset$.

Note that for any abstract expression A, $\mathit{size}([\![A]\!]\rho)$ is bounded by some polynomial $P(n)$.

The abstract expressions enjoy the following properties, which allow us to prove that transitive closure is not efficiently expressible in $\mathcal{NRA}(\mathit{powerset})$:

1. Abstract expressions are closed under application of functions in $\mathcal{NRA}$ (lemma 5.1).
2. r_n can be expressed as an abstract expressions, but its transitive closure $tc(r_n)$ cannot (see lemma 5.3).
3. Any set expressed by an (even open) abstract expression has either $O(1)$ or $\Omega(n)$ elements, and, hence, any function in $\mathcal{NRA}(\mathit{powerset})$ applied to some abstract expression is either another abstract expression or has exponential complexity (lemma 5.8).

5.2 Evaluation of Abstract Expressions

The key lemma is given below:

Lemma 5.1 (Evaluation lemma) *Let A be some (not necessarily closed) abstract expression of type s, and $f \in \mathcal{NRA}$. Then there is some abstract expression A' such that $f(A) \Downarrow A'$, meaning that $\forall n, \forall \rho$, $f([\![A]\!]\rho) \Downarrow [\![A']\!]\rho$.*

Proof. We prove it straightforward, by induction on the structure of f. We illustrate some of the cases:

Case $g \circ f$. Apply induction on $f(A)$ to get $f(A) \Downarrow A'$, next, apply induction on $g(A')$, to get $g(A') \Downarrow A''$. Clearly, $(g \circ f)(A) \Downarrow A''$.

Case $map(f)$ To compute $map(f)(\{A \mid \vec{x} = 0, n\})$, compute first $f(A) \Downarrow A'$ (i.e. apply induction hypothesis), and get $map(f)(\{A \mid \vec{x} = 0, n\}) \Downarrow \{A' \mid \vec{x} = 0, n\}$. The other cases are treated as follows: $map(f)(A_1$ when $C_1; \ldots; A_l$ when $C_l) = map(f)(A_1)$ when $C_1; \ldots; map(f)(A_l)$ when C_l, and $map(f)(A_1 \cup A_2) = map(f)(A_1) \cup map(f)(A_2)$.

Case $=$ This is the case where we are forced to introduce guarded expressions. E.g. $= (e, e') \Downarrow (true$ when $e = e'; false$ when $e \neq e')$.

Case *empty* One can show that for any abstract expression A, there is some condition C_A on its free variables expressing the fact that A is defined. Then $empty(\{A \mid \vec{x} = 0, n\}) \Downarrow false$ when $\exists \vec{x}.C_A$; *true* when $\neg(\exists \vec{x}.C_A)$. So it suffices to observe that the conditions enjoy a quantifier elimination property, i.e. $\exists \vec{x}.C_A$ is equivalent to some quantifier free condition C'_A.

□

We shall generalize this lemma to $\mathcal{NRA}(powerset)$ (see lemma 5.8), showing that $f(A)$ is either some abstract expression, or requires exponential space to compute. The key step will be to show that some set $\{A \mid \vec{x} = 0, n\}$ has either $O(1)$ or $\Omega(n)$ elements. The problem in proving that is that some elements $A(\vec{x}, \vec{y})$ and $A(\vec{x}', \vec{y})$ may be equal, even when $\vec{x} \neq \vec{x}'$ (here $\vec{y}$ are the other free variables in A, besides $\vec{x}$). In order to count the distinct elements, let $\vec{x}'$ be a fresh tuple of variables, and $A' = A[\vec{x}'/\vec{x}]$, i.e. A with the variables $\vec{x}$ substituted by $\vec{x}'$. The inequality $A \neq A'$ can be expressed as a simple condition $D(\vec{x}, \vec{x}', \vec{y})$: whenever $\vec{x}$ and $\vec{x}'$ satisfy D, $A(\vec{x}, \vec{y})$ and $A(\vec{x}', \vec{y})$ are distinct elements of the set. So we are left to prove that some condition $D(\vec{x}, \vec{x}', y)$ is satisfied by $O(1)$ or $\Omega(n)$ distinct $\vec{x}$'s.

5.3 Affine and Variable Affine Spaces

We are interested in the satisfiability of conditions for n is large enough. Thus, we say that some condition $C(\vec{x})$, where $\vec{x} = (x_1, \ldots, x_k)$ is **satisfiable**, if it is satisfiable in the classical sense for n large enough, i.e. iff $\exists n_0 > 0, \forall n \geq n_0, \exists \vec{x} \in [n]^k$ such that $C(\vec{x})$ is true.

We shall concentrate on **conjunctive conditions**, defined to be conjuncts of simple conditions. A convenient way to think of satisfiable conjunctive conditions is to view them as *affine spaces*. We define an **affine space** to be a subset U of $[n]^k$, of the form $\{\vec{e}(\vec{\alpha}) \mid \alpha \in [n]^p, \Gamma(\vec{\alpha})\}$, where $\vec{e}(\vec{\alpha})$ is a vector of simple expressions whose free variables are exactly $\vec{\alpha}$, and $\Gamma(\vec{\alpha})$ is a conjunction of negative simple conditions. p is called **the dimension** of U, and the variables $\vec{\alpha}$ are called the **parameters** of the affine space. The affine spaces enjoy the following properties:

Proposition 5.2 *1. For any satisfiable conjunctive condition $C(\vec{x})$, $U = \{\vec{x} \mid C(\vec{x})\}$ is an affine space. Conversely, any affine space U is of this form, for some satisfiable conjunctive condition $C(\vec{x})$.*

2. An affine spaces U of dimension p has $n^p - O(n^{p-1})$ $(= \Theta(n^p))$ elements. Hence, an affine space of dimension 0 has exactly 1 element. As a consequence, any affine space is nonempty.

3. The intersection of two affine spaces $U \cap U'$ is either empty or another affine space

As a consequence we have:

Corollary 5.3 *No abstract expression can denote $tc(r_n)$ for all $n > 0$.*

Proof. Indeed, $tc(r_n)$ must have $\frac{n(n-1)}{2}$ elements. But one can prove that any closed abstract expression of type $\{\mathbb{N} \times \mathbb{N}\}$ denotes a union of affine spaces: none of them can have dimension 2 (else we get $n^2 - O(n)$ elements), so their union has at most $O(n)$ elements, and it cannot denote $tc(x_n)$. $\square$

With some care, we can view a satisfiable conjunctive condition $C(\vec{x}, \vec{y})$ as a *variable affine space* $V(\vec{y})$. Namely, we define a **variable affine space** to be a set $V(\vec{y}) = \{\vec{e}(\vec{\alpha}, \vec{y}) \mid \vec{\alpha} \in [n]^p, \Gamma(\vec{\alpha}, \vec{y})\}$, in which $\Gamma(\vec{\alpha}, \vec{y})$ is a conjunction of negative conditions.

Example 5.4 *$U_1 = \{(3, \alpha_1 - 5, \alpha_2, \alpha_1) \mid \vec{\alpha} \in [n]^2\}$ is an affine space of dimension 2; it corresponds to the condition $C(x_1, x_2, x_3, x_4) = (x_1 = 3 \wedge x_2 = x_4 - 5)$. $U_2 = \{(n-3, \alpha_1, \alpha_2, \alpha_3) \mid \vec{\alpha} \in [n]^3 \wedge \alpha_1 \neq \alpha_2 \wedge \alpha_1 \neq \alpha_3 + 5\}$ is an affine space of dimension 3. Also, $U_3(y) = \{(\alpha + 2, y - 1) \mid \alpha \in [n] \wedge \alpha \neq n \wedge \alpha \neq y - 3 \wedge y \neq 1\}$ is a variable affine space, of dimension 1, which is empty when $y = 1$.*

The properties mentioned in proposition 5.2 can be extended to variable affine space. We only state this for item 1:

Proposition 5.5 *For any satisfiable conjunctive condition $C(\vec{x}, \vec{y})$, there are some affine space U and some variable affine space $V(\vec{y})$ such that $C(\vec{x}, \vec{y})$ is equivalent to $\vec{y} \in U \wedge \vec{x} \in V(\vec{y})$, and, for n large enough, $\forall y \in U, V(\vec{y}) \neq \emptyset$. The converse is also true.*

For some (variable) affine space $U(\vec{y})$ and dimension i, if $e_i(\vec{\alpha})$ depends on some parameter α, we say that $U(\vec{y})$ is **free** along the dimension i. Else (if $e_i(\vec{\alpha})$ is constant or depends on some $\vec{y}$), we say that $U(\vec{y})$ is **bound** at the dimension i. In the above example, both U_1 and U_2 are free along their dimensions 2,3 and 4, and bound along dimension 1. $U_3(y)$ is free along dimension 1 and bound along dimension 2.

5.4 The Powerset of Abstract Expressions

Let $D(\vec{x}, \vec{x}', \vec{y})$ be some condition, where $\vec{x}, \vec{x}'$ have the same length p, and $\vec{y}$ has length q. We think of D as defining, for each $\vec{y} \in [n]^q$, a binary relation on $[n]^p$, namely $\vec{x}$ and $\vec{x}'$ are related iff $D(\vec{x}, \vec{x}', \vec{y})$. Now consider $\vec{y} \in [n]^q$ and a sequence of m different values in $[n]^p$, $\vec{x}_1, \ldots, \vec{x}_m$. We say that this sequence **is included in** D **for** $\vec{y}$, iff $D(\vec{x}_i, \vec{x}_j, \vec{y})$, forall $1 \leq i < j \leq m$.

Lemma 5.6 *Let $D(\vec{x}, \vec{x}', \vec{y})$ be a conjunctive condition as described above, such that for arbitrarily large n there is some $\vec{y} \in [n]^q$ and a sequence of $m = 4$ distinct vectors $\vec{x}_1, \vec{x}_2, \vec{x}_3, \vec{x}_4$ included in D for $\vec{y}$. Then there is some affine space $U \subseteq [n]^q$ such that for any $\vec{y} \in U$ there is some sequence of length $m = \Omega(n)$ included in D for $\vec{y}$.*

Proof. Obviously D is satisfiable, in the sense of section 5.3. We start by splitting D into $D(\vec{x}, \vec{x}', \vec{y}) = E(\vec{x}, \vec{x}') \wedge F(\vec{x}, \vec{y}) \wedge F'(\vec{x}', \vec{y})$, such that $E(\vec{x}, \vec{x}')$ contains exactly those conditions which mention *both* one variable from $\vec{x}$ and one from $\vec{x}'$. The conditions mentioning only variables in $\vec{y}$ can be included arbitrarily in F or F'.

Now consider the condition $G(\vec{x}, \vec{y}) \stackrel{\text{def}}{=} F(\vec{x}, \vec{y}) \wedge F'(\vec{x}, \vec{y})$ (i.e. we substitute $\vec{x}'$ with $\vec{x}$ in F'). We observe that both $G(\vec{x}_2, \vec{y})$ and $G(\vec{x}_3, \vec{y})$ are true, hence, G is satisfiable, in the sense of section 5.3. By proposition 5.5, $G(\vec{x}, \vec{y})$ is equivalent to $\vec{y} \in U \wedge \vec{x} \in V(\vec{y})$, for some affine space U and variable affine space $V(\vec{y}) = \{\vec{e}(\vec{\alpha}, \vec{y}) \mid \Gamma(\vec{\alpha}, \vec{y})\}$ of nonzero dimension. More, $\vec{x}_2, \vec{x}_3 \in V(\vec{y})$ and $\forall \vec{y} \in U, \forall \vec{x}, \vec{x}' \in V(\vec{y})$, $F(\vec{x}, \vec{y}) \wedge F'(\vec{x}', \vec{y})$ is true.

Next we take care of the condition $E(\vec{x}, \vec{x}')$, which we split into *positive* and *negative* simple conditions, $E(\vec{x}, \vec{x}') = E_P(\vec{x}, \vec{x}') \wedge E_N(\vec{x}, \vec{x}')$. First we prove that $E_P(\vec{x}_2, \vec{x}_2)$ and $E_P(\vec{x}_3, \vec{x}_3)$ are both true. Indeed, consider some condition $x_i = x'_j + c$ in E_P. Since $E_P(\vec{x}_1, \vec{x}_3)$, $E_P(\vec{x}_2, \vec{x}_3)$ and $E_P(\vec{x}_1, \vec{x}_2)$ are true, we conclude that x_2 satisfies the condition $x_i = x_j + c$, and, in fact, that $E_P(\vec{x}_2, \vec{x}_2)$ is true. Similar, $E_P(\vec{x}_3, \vec{x}_3)$ is true.

We process, one by one, all conditions in E_P, restricting U and $V(\vec{y})$ for each of them, such as to guarantee that $\forall \vec{y} \in U, \forall \vec{x}, \vec{x}' \in V(\vec{y}), E_P(\vec{x}, \vec{x}')$ holds. Let $x_i = x'_j + c$ be such a condition. Four cases can occur:

(1) $V(\vec{y})$ is *bound* along the dimensions i and j (see section 5.3). (2) $V(\vec{y})$ is free along one dimension and bound along the other. (3) $V(\vec{y})$ is free along both dimensions i and j, and has the same parameter variable α in both places (this includes the case $i = j$). (4) $V(\vec{y})$ is free along both dimensions i and j, and has different parameter variables at these dimensions. In cases (1), (2) and (3) we add the condition $e_i = e_j + c$, which in case (1) may restrict U, in case (2) restricts $V(\vec{y})$, and in case (3) is a tautology. In case (4), for some particular n, let v be the value of position i in $\vec{x}_1, \vec{x}_2, \vec{x}_3$ (one can see that all of them must have the same value on position i). The we impose the condition $e_i = v \wedge e_j = v - c$, which affects only $V(\vec{y})$. In all four cases, we can prove that the resulting U and $V(\vec{y})$ satisfy $U \neq \emptyset$ and $V(\vec{y})$ has a nonzero dimension (because $V(\vec{y})$ has at least two elements, $\vec{x}_2, \vec{x}_3$). More, $V(\vec{y})$ has constants (not $\vec{\alpha}$'s or $\vec{y}$'s) on both dimensions i and j, hence $\forall \vec{y} \in U$, any two $\vec{x}, \vec{x}' \in V(\vec{y})$ will satisfy the condition $x_i = x'_j + c$.

Finally consider, one by one, each of the negative conditions in E_N, say $x_i \neq x'_j + c$. We distinguish the same four cases as above. Cases (1) and (2) are treated similarly, but without decreasing the dimensions of U and $V(\vec{y})$. For the cases (3) and (4), the condition $x_i \neq x'_j + c$ translates to $\alpha_l \neq \alpha'_{l'} + c'$, where l and l' are equal in case (3) and different in case (4). Here, we just "remeber" the constant $| c' |$. Let γ be the largest such constant. In the end, we get some affine space U and variable affine space $V(\vec{y}) = \{\vec{e}(\vec{\alpha}, \vec{y}) \mid \Gamma(\vec{\alpha}, \vec{y})\}$, with the same dimension $r > 0$ as the previous one, with the property that for any $\vec{y} \in U$ and any $\vec{x}, \vec{x}' \in V(\vec{y})$, such that all parameters for x and x' differ by more than γ (i.e. $| \alpha_l - \alpha'_{l'} | > \gamma$, for all l, l'), then $D(\vec{x}, \vec{x}', \vec{y})$ is true.

Since $\Gamma(\vec{\alpha}, \vec{y})$ contains only negative conditions, one can find, for every n sufficiently large and any $\vec{y} \in U$, some $\vec{\alpha}$ satisfying $\Gamma(\vec{\alpha}, \vec{y})$, whose elements are "small", i.e. $\alpha_l \leq \delta$, $\forall l$, where δ is independent of n. Then, obviously, there is a sequence of length $\frac{n-\delta}{(\gamma+\delta+1)}$ included in D for $\vec{y}$. □

Next, we make use of the following lemma:

Lemma 5.7 *([5], pp. 104, theorem 1) Let G be a complete, undirected graph with $C_{2m-2}^{m-2} (= \frac{(2m-2)!}{(m-1)!})$ vertices, whose edges have been colored with red or blue. Then there is a complete subgraph with m vertices having all edges colored with the same color.*

As a consequence, if some disjunction $D = D_1 \vee \ldots \vee D_k$ includes a "long" sequence, then at least one of its D_l includes a sequence of length m. (Here "long" could be m_k, where $m_1 = m$, $m_{i+1} = C_{2m_i-2}^{m_i-1}$.)

Now we can generalize lemma 5.1 for $\mathcal{NRA}(powerset)$.

Lemma 5.8 *Let A be some (not necessarily closed) abstract expression of type s, C some condition and $f : s \to t$ in $\mathcal{NRA}(powerset)$. Then one of the following holds:*

1. *There is some abstract expression A' such that, for any ρ satisfying C, we have $f([\![A]\!]\rho) \Downarrow [\![A']\!]\rho$. We abbreviate: $f(A) \Downarrow A'$ WHEN C. More, there is some approximation f_m of f such that $f_m(A) \Downarrow A'$ WHEN C.*
2. *The complexity of $f(A) \Downarrow$ WHEN C is $\Omega(2^{cn})$, for some $c > 0$.*

Proof. The proof is done by induction on the structure of f. All cases, except *powerset*, are straightforward extensions of those in lemma 5.1. E.g. for the case $(g \circ f)(A) \Downarrow$ WHEN C, we have to argue, in addition, that, if $f(A) \Downarrow$ WHEN C has exponential complexity, then so has $(g \circ f)(A) \Downarrow$ WHEN C. Otherwise, $f(A) \Downarrow$

A' WHEN C, and we argue similarly for $g(A') \Downarrow$. If the latter doesn't have exponential complexity either, then, by induction, we also get numbers m, m' s.t. $f_m(A) \Downarrow A'$ WHEN C and $g_{m'}(A') \Downarrow A''$ WHEN C. Pick $m = \max(m, m')$.

Note that the condition C is enriched when computing $f(A)$, with A a guarded expression, or with f a *if then else* function.

So it remains to consider the case *powerset*. We only look at the case when the abstract expression is of the form $\{A \mid \vec{x} = 0, n\}$ (the other two cases, $A \cup A'$ or the guarded expression case, are reduced to this one). Then, it suffices to prove that one of the following cases must occur:

1. There is some number m, independent of n (dependent only on C and A), such for any n and for any $\vec{y}$ satisfying $C(\vec{y})$, the set $\{A \mid \vec{x} = 0, n\}$ has at most m elements. More, in this case we can actually find abstract expressions $A_1, \ldots, A_m$ naming these at most m elements. In this case $powerset(\{A \mid \vec{x} = 0, n\}) \Downarrow A'$, were A' is just the set of all 2^m subsets of $\{A_1, \ldots, A_m\}$. Obviously, in this case f is equivalent to the m'th approximation of *powerset*, i.e. $powerset(\{A \mid \vec{x} = 0, n\}) = powerset_m(\{A \mid \vec{x} = 0, n\})$, under the condition C.
2. For every n, there is some environment ρ satisfying $C(\vec{y})$, such that the set $[\![\{A \mid \vec{x} = 0, n\}]\!]\rho$ contains at least $\Omega(n)$ distinct elements. Then condition 2 holds.

Suppose the first condition does not hold, i.e. for any $m > 0$, we can find n and ρ (an environment satisfying C) under which the set has $\geq m$ elements. Let A' be $A[\vec{x}'/\vec{x}]$, i.e. fresh variables $\vec{x}'$ are substituted for $\vec{x}$. Recall that equality is definable at all types in $\mathcal{NRA}$ (proposition 2.1). We evaluate $= (A, A')$, using lemma 5.1: the result is of type $\mathbf{B}$, so it must have the form *false* when $D_1(\vec{x}, \vec{x}', \vec{y})$; *true* when $D_2(\vec{x}, \vec{x}', \vec{y})$. Consider the condition $D(\vec{x}, \vec{x}', \vec{y}) \stackrel{\text{def}}{=} C(\vec{y}) \wedge D_1(\vec{x}, \vec{x}', \vec{y})$: for any $m > 0$, there is some $\vec{y}$ and a sequence of length m included in D for $\vec{y}$. Writing D in conjunctive normal form, by lemma 5.7, at least one of is conjuncts D' satisfies the conditions of lemma 5.6. For D', we apply lemma 5.6, and get an affine space U, such that for any ρ assigning values in U to $\vec{y}$, we have (1) C is true, and (2) $[\![\{A \mid \vec{x} = 0, n\}]\!]\rho$ has $\Omega(n)$ elements. Hence, the complexity of $f(\{A \mid \vec{x} = 0, n\}) \Downarrow$ is $\Omega(2^{cn})$. □

Now we are ready to prove our main results, which we restate here:

Theorem 4.1 *For any function $f \in \mathcal{NRA}(powerset)$ of type $f : \{\mathbb{N} \times \mathbb{N}\} \to \{\mathbb{N} \times \mathbb{N}\}$ such that $f(r_n) \Downarrow q_n$ for every $n > 0$, the complexity of $f(r_n) \Downarrow$ is $\Omega(2^{cn})$, for some $c > 0$.*

Proposition 4.2 *For any function $f : \{\mathbb{N} \times \mathbb{N}\} \to s$ in $\mathcal{NRA}(powerset)$, either there exists some approximation f_m of f such that $f_m(r_n) = f(r_n)$, $\forall n > 0$, or the complexity of $f(r_n) \Downarrow$ is $\Omega(2^{cn})$, for some $c > 0$.*

Proof. Let $f \in \mathcal{NRA}(powerset)$, $f : \{\mathbb{N} \times \mathbb{N}\} \to s$ be such that the complexity of $f(r_n) \Downarrow$ is not $\Omega(2^{cn})$. Taking $C = true$, only case 1 of lemma 5.8 can hold, hence there is some approximation f_m of f performing the same computation on r_n (which proves proposition 4.2), and there is some abstract expression denoting its result; by corollary 5.3, its result cannot be $tc(r_n)$, which proves theorem 4.1. □

6 Conclusions

We have proven that transitive closure, although known to be expressible with *powerset*, is not expressible with *powerset* in an *efficient* way. We conjecture that *any* query which can be computed efficiently with *powerset*, can also be computed without *powerset*, but this problem remains open. It would prove that 'we cannot use the powerset in any tractable way'.

7 Acknowledgments

We would like to thank Val Breazu-Tannen and Peter Buneman for their suggestions and support.

References

[1] Serge Abiteboul and Catriel Beeri. On the power of languages for the manipulation of complex objects. In *Proceedings of International Workshop on Theory and Applications of Nested Relations and Complex Objects*, Darmstadt, 1988. Also available as INRIA technical report 846.

[2] Serge Abiteboul and Victor Vianu. Generic computation and its complexity. In *Proceedings of 23rd ACM Symposium on the Theory of Computing*, 1991.

[3] Alfred V. Aho and Jeffrey D. Ullman. Universality of data retrieval languages. In *Proceedings 6th Symposium on Principles of Programming Languages, Texas, January 1979*, pages 110–120, 1979.

[4] David Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[5] Bela Bollobas. *Graph Theory: An Introductory Course.* Springer-Verlag, 1979.

[6] Val Breazu-Tannen, Peter Buneman, and Limsoon Wong. Naturally embedded query languages. In J. Biskup and R. Hull, editors, *LNCS 646: Proceedings of 4th International Conference on Database Theory, Berlin, Germany, October, 1992*, pages 140–154. Springer-Verlag, October 1992. Available as UPenn Technical Report MS-CIS-92-47.

[7] J. Hopcroft and J. Ullman. *Introduction to automata theory, languages, and computation.* Addison-Wesley, 1979.

[8] Neil Immerman. Languages that capture complexity classes. *SIAM Journal of Computing*, 16:760–778, 1987.

[9] Neil Immerman. Expressibility and parallel complexity. *SIAM Journal of Computing*, 18:625–638, 1989.

[10] Gilles Kahn. Natural semantics. In *Proceedings of Symposium on Theoretical Aspects of Computer Science*, pages 22–39. Springer-Verlag, 1987.

[11] Jan Paredaens and Dirk Van Gucht. Possibilities and limitations of using flat operators in nested algebra expressions. In *Proceedings of 7th ACM Symposium on Principles of Database Systems,Austin, Texas*, pages 29–38, 1988.

[12] Jan Paredaens and Dirk Van Gucht. Converting nested relational algebra expressions into flat algebra expressions. *ACM Transaction on Database Systems*, 17(1):65–93, March 1992.

[13] H.-J. Schek and M. H. Scholl. The relational model with relation-valued attributes. *Information Systems*, 11(2):137–147, 1986.

[14] S. J. Thomas and P. C. Fischer. Nested relational structures. In P. C. Kanellakis and F. P. Preparata, editors, *Advances in Computing Research: The Theory of Databases*, pages 269–307, London, England, 1986. JAI Press.

[15] Limsoon Wong. Normal forms and conservative properties for query languages over collection types. In *Proceedings of 12th ACM Symposium on Principles of Database Systems*, pages 26–36, Washington, D. C., May 1993. Full paper available as UPenn Technical Report MS-CIS-92-59.