THE POWERSET ALGEBRA AS A RESULT
OF ADDING PROGRAMMING CONSTRUCTS
TO THE NESTED RELATIONAL ALGEBRA

Marc Gyssens, University of Antwerp (UIA), B-2610 Antwerpen, Belgium

Dirk Van Gucht, Indiana University, Bloomington, IN 47405-4101, USA

In this paper, we discuss augmentations of the nested relational algebra with programming constructs, such as while-loops and for-loops We show that the algebras obtained in this way are equivalent to a slight extension of the powerset algebra, thus emphasizing both the strength and the naturalness of the powerset algebra as a tool to manipulate nested relations, and, at the same time, indicating more direct ways to implement this algebra

1. Introduction

In the last years, much attention has been paid to structured relations In order to model some database applications more naturally, Makinouchi proposed to generalize the relational model by removing Codd's first normal form assumption [7], thus allowing relations with set-valued attributes [16] Subsequently, a generalization of the relational algebra to relations with set-valued attributes was introduced by Jaeschke and Schek [14] More specifically, they presented the nest and the unnest operator as tools to restructure such relations Finally, Thomas and Fischer generalized this model by allowing nested relations of arbitrary depth [21] Calculus like query languages for such models were defined, e g in [1,17,19]

In an algebraic language used in [15], Kuper and Vardi introduced the powerset operator Recently, much attention has been paid to the powerset algebra obtained by adding this operator to the nested algebra (e g [1,10,13]) This algebra was found to be considerably more expressive than the nested algebra [10,12] Another extension of the nested relational algebra that has been considered, is its least fixpoint closure [2,10], which is equivalent to the powerset algebra

In [4], Chandra and Harel introduced a very powerful query language for classical relations, called QL In [5], they discussed RQL, which is a restriction of QL Both languages contain programming constructs, such as while-loops They showed that RQL is at least as expressive as the least fixpoint closure of relational query languages They furthermore observed that showing

that RQL is equivalent in expressive power to the fixpoint closure of a relational query language would imply that $PTIME = PSPACE$, which is generally believed to be false

In this paper, we add programming constructs such as while-loops and for-loops to the nested algebra and show that these extensions yield a query language equivalent to the powerset algebra and hence also the least fixpoint closure of the nested relational algebra, thus underlining the strength and the naturalness of this algebra, establishing a sharp distinction between the properties of query languages for the nested relational model and the standard relational model, and, finally, indicating ways to implement the powerset algebra

2. A model for working with nested relations

2.1. Nested relations

Basically we assume that we have an infinitely enumerable set U of *elementary attributes* and an infinitely enumerable set V of *elementary values* In this section, we explain how arbitrary attributes and values, relation schemes, relation instances and relations are constructed from these

First, we define an attribute Attributes can either be elementary or composed The latter ones are sets of attributes (which can be composed in turn), the values associated to them are relation instances over that set of attributes, interpreted as a scheme

Definition 2 1 1

The set of all attributes $\mathcal{U}$ is the smallest set containing U such that for each finite subset X of $\mathcal{U}$ in which no elementary attribute appears more than once, $X \in \mathcal{U}$ ∎

An attribute of U is called an *elementary attribute*, an attribute of $\mathcal{U} - U$ is called a *composed attribute*

The values associated to composed attributes will be called quite fittingly composed values The definition of a *relation scheme* is now very straightforward

Definition 2 1 2

A relation scheme Ω is a composed attribute, i e an element of $\mathcal{U} - U$　∎

Example 2 1 1

Consider a relation representing persons, their jobs and the cities in which these jobs are executed

PERSON	{{JOB	} CITY }
Jeff Willows	professor president	Austin
	consultant	Dallas
Mary Higgins		

There are two levels of nesting jobs are grouped by the city in which they are executed and pairs of sets of jobs and cities are grouped by the person having these jobs The scheme of the above relation is a set of two attributes, the former of which is the elementary attribute *PERSON* whereas the latter is the composed attribute $\{\{JOB\}, CITY\}$

Let us now explain on this example, how we intend to define relation instances The instance of the above relation will consist of two tuples In each of these tuples, the value corresponding to *PERSON* is atomic whereas the composed value corresponding to $\{\{JOB\},$ $CITY\}$ is in turn a relation instance over the scheme $\{\{JOB\}, CITY\}$ In the former tuple, this relation instance again consists of two tuples In each of these tuples, the value corresponding to $\{JOBS\}$ is a relation instance over $\{JOBS\}$ whereas the value corresponding to *CITY* is atomic In the latter tuple the aforementioned relation instance is empty　∎

It should be clear by now that the notions of value, tuple and instance are so closely intertwined that it is easier to define them jointly

Definition 2 1 3

The set $\mathcal{V}$ of all values, the set $\mathcal{I}_X$ of all instances over $X \in \mathcal{U} - U$, the set $\mathcal{T}_X$ of all tuples over $X \in \mathcal{U} - U$ and the set $\mathcal{I}$ of all instances are the smallest sets satisfying
1 $\mathcal{V} = V \cup \mathcal{I}$,
2 $\mathcal{I} = \bigcup_{X \in \mathcal{U}-U} \mathcal{I}_X$,
3 $\mathcal{I}_X$ *consists of all finite subsets of $\mathcal{T}_\lambda$,*
4 $\mathcal{T}_X$ *consists of all mappings t from X into $\mathcal{V}$, called tuples, satisfying $t(A) \in V$ for all elementary attributes $A \in X \cap U$ and $t(Y) \in \mathcal{I}_Y$ for all composed attributes $Y \in X - U$*　∎

We now have all the necessary ingredients to formally define a relation

Definition 2 1 4

A relation is a pair (Ω, ω) where $\Omega \in \mathcal{U} - U$ and $\omega \in \mathcal{I}_\Omega$ Ω is called the scheme of the relation and ω is called the instance of the relation If $\Omega \subseteq U$, then (Ω, ω) is called a flat relation　∎

Now let $\mathcal{W}$ be a set of relation instances over Ω For notational convenience, we shall denote by $(\{\Omega\}, \mathcal{W})$ the relation with scheme $\{\Omega\}$ and instance $\{t \in \mathcal{T}_{\{\Omega\}} \mid t(\Omega) \in \mathcal{W}\}$ Note that there is no essential difference between a relation instance over a singleton scheme consisting of one composed attribute, and a set of relation instances over that composed attribute regarded as a scheme We shall make thankful use of this dualism in the sequel

2.2. The nested relational algebra

In this section, we define a nested algebra based on the model for relations described in the previous section It is generated by eight operators Basically, these operators are borrowed from the classical "flat" relational algebra, except for the nesting and unnesting However, some technicalities were unavoidable to fit them in into our formalism

Definition 2 2 1

Let (Ω, ω), (Ω, ω_1), (Ω, ω_2), (Ω_1, ω_1) and (Ω_2, ω_2) be relations Suppose that the sets of elementary attributes from which Ω_1 and Ω_2 are built, are disjoint

- *The union $(\Omega, \omega_1) \cup (\Omega, \omega_2)$ equals $(\Omega, \omega_1 \cup \omega_2)$,*
- *The difference $(\Omega, \omega_1) - (\Omega, \omega_2)$ equals $(\Omega, \omega_1 - \omega_2)$,*
- *The cartesian product $(\Omega_1, \omega_1) \times (\Omega_2, \omega_2)$ equals (Ω', ω') where $\Omega' = \Omega_1 \cup \Omega_2$ and*

$$\omega' = \{t \in \mathcal{T}_{\Omega'} \mid t|_{\Omega_1} \in \omega_1 \ \& \ t|_{\Omega_2} \in \omega_2\}$$

- *Let $\Omega' \subseteq \Omega$ The projection $\pi_{\Omega'}(\Omega, \omega)$ equals (Ω', ω') where $\omega' = \{t|_{\Omega'} \mid t \in \omega\}$,*
- *Let $X \subseteq \Omega$ The nesting $\nu_X(\Omega, \omega)$ equals (Ω', ω') where $\Omega' = (\Omega - X) \cup \{X\}$ and*

$$\omega' = \{t \in \mathcal{T}_{\Omega'} \mid \exists t' \in \omega \ t|_{\Omega-X} = t'|_{\Omega-X} \ \& $$
$$t(X) = \{t''|_X \mid t'' \in \omega \ \& \ t'|_{\Omega-X} = t''|_{\Omega-\lambda}\}\}$$

- *Let $X \in \Omega - U$ The unnesting $\mu_X(\Omega, \omega)$ equals (Ω', ω') where $\Omega' = (\Omega - \{X\}) \cup X$ and*

$$\omega = \{t \in \mathcal{T}_{\Omega'} \mid \exists t' \in \omega \ t|_{\Omega-\{X\}} = t'|_{\Omega-\{X\}}$$
$$\& \ t|_X \in t'(X)\}$$

Let (Ω, ω) be a relation scheme Let φ be a permutation on U φ is extended in the natural way to $\mathcal{U}$, to $\mathcal{I}$ and to $\mathcal{V}$
- *The renaming $\rho^\varphi(\Omega, \omega)$ equals $(\varphi(\Omega), \varphi(\omega))$,*

226

- *Assume furthermore that $\varphi(\Omega) = \Omega$ The selection $\sigma^\varphi(\Omega, \omega)$ equals (Ω, ω') where*

$$\omega' = \{t \in \omega \mid \forall X \in \Omega \ \varphi(t(X)) = t(\varphi(X))\} \quad \blacksquare$$

Example 2 2 1

Reconsider the relation in Example 2 1 1 If we denote this relation by (Ω, ω), then $\mu_{\{\{JOB\}, CITY\}}(\Omega, \omega)$ yields a relation with scheme $\Omega' = \{PERSON, \{JOB\}, CITY\}$ which can be represented as

PERSON	{JOB }	CITY
Jeff Willows	professor president	Austin
Jeff Willows	consultant	Dallas

If we nest this last relation over $\{\{JOBS\}, CITY\}$, we again obtain a relation over Ω the instance of which corresponds to the first tuple of ω Note that in general, an unnesting cannot be undone by a nesting, even if no empty composed values are present A nesting on the other hand, can always be undone by the corresponding unnesting $\quad \blacksquare$

Note that the cartesian product is only defined for relations with completely "independent" schemes This is actually not a severe restriction it is indeed always possible to arrange that the schemes of two relations have no elementary attributes in common by performing an appropriate renaming

We end this discussion about the basic nested algebra operators with a notational issue In most practical cases, renaming involves only one attribute X at the time If X is renamed to X', and if, in case X and X' are composed attributes, no ambiguity is possible as to how the renaming is done, we shall denote this operation as $\rho_{X' \leftarrow X}$ We shall use the same notation if X is a set of attributes of the scheme under consideration, and each attribute of X is renamed in a well known way to an attribute of X' Similarly, if selection comes down to only checking whether the values for composed attributes X and X' are equal upon renaming and if no ambiguity is possible as to how the elementary attributes in X and X' are to be matched, we shall denote this selection by $\sigma_{X=X'}$ Again, we shall use the same notation if X and X' are sets of attributes of the scheme under consideration

We can now define a *nested algebra expression (nae)*

Definition 2 2 2

1 *The variables $x, y, z, \quad$ are naes,*
2 *For all $\Omega \in \mathcal{U} - U$, $(\Omega, \emptyset)$ is a nae,*
3 *For all $\Omega \in \mathcal{U} - U$, $(\{\Omega\}, \{\emptyset\})$ is a nae,*
4 *For all naes, the basic operators of Definition 2 2 1 applied to them, are also naes, provided these new expressions make sense* $\quad \blacksquare$

We implicitly assume that all variables are typed, i e associated to relations having one particular scheme However, we shall not explicitly use this typedness for convenience of notation The expressions introduced in items 1, 2 and 3 of Definition 2 2 2 will be called *primitive expressions*

The set of all naes will be denoted by $\mathcal{N}$ If $E(x, y, \quad) \in \mathcal{N}$ and $r, s, \quad$ are a finite sequence of relations the schemes of which are "compatible" with the variables of the expression, then $E(r, s, \quad)$ is interpreted as the relation obtained by substituting every occurrence of a variable in $E(x, y, \quad)$ by the corresponding relation Obviously, the scheme of $E(r, s, \quad)$ does not depend on the actual instances of $r, s, \quad$ Therefore we shall often denote this scheme as Ω^E

To avoid extensive use of brackets, we assume the following precedence on nested algebra operators unary operators, cartesian product, set operators

3. The powerset algebra

Recently, much attention has been paid to the expressiveness of the nested relational algebra [1,13,18] In order to deal with this problem, it suffices to consider single relations only, since a database can always be represented as the cartesian product of its non-empty members In its most general form, the question that must be asked, is [4,5] *Let Q be a computable query, i e a partial recursive mapping from relations to relations that preserves isomorphism Does there exist $E(x) \in \mathcal{N}$ such that $\forall r \ E(r) = Q(r)$?* Although it has been shown [8,22] that it is always possible to find an expression that satisfies this equality for any particular relation, there is in general no expression that will do for all relations E g the transitive closure of a binary flat relation, which is not expressible in the classical relational algebra, is also not expressible in the nested relational algebra [18]

Therefore, several attempts have been made to enrich the nested relational algebra One of these consists of adding the powerset operator to the nested algebra This operator was introduced by Kuper and Vardi in [15] as one of the primitive operators in their algebraic query language for database logic Basically the powerset operator generates all subsets of a given relation

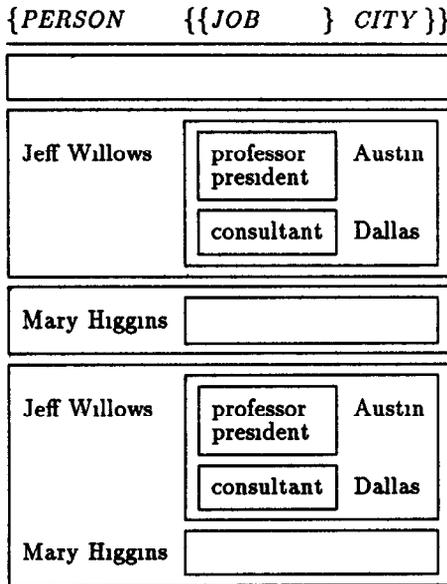## Definition 3 1

Let (Ω, ω) be a relation Let 2^ω denote the set of all subsets of ω Then (using the notation introduced earlier) the powerset $\Pi(\Omega, \omega)$ equals $(\{\Omega\}, 2^\omega)$ $\quad \blacksquare$

Example 3 1

Reconsider the relation (Ω, ω) represented in Example 2 1 1 $\Pi(\Omega, \omega)$ is a relation with scheme $\{\{PERSON,$

$\{\{JOB\}, CITY\}\}\}$ which can be represented as

$\{PERSON \quad \{\{JOB \quad \} CITY \}\}$

Jeff Willows	professor president	Austin
	consultant	Dallas
Mary Higgins		
Jeff Willows	professor president	Austin
	consultant	Dallas
Mary Higgins		

Using a combinatorial argument, it was established in [10] (a similar result was obtained independently by Houben, Paredaens and Tahon [12])

Theorem 3 1
There is no nae that expresses the powerset operator.

We now consider the *powerset algebra* whose set of expressions $\mathcal{P}$ is generated by the basic operators of the nested relational algebra defined in Definition 2 2 1, augmented with the powerset operator. An expression of the powerset algebra is called a *powerset algebra expression (pae)*. The expressiveness of the powerset algebra is illustrated in part by a result of [10]

Theorem 3 2
In the powerset algebra, either the nest operator or the difference can be considered as redundant

4. The least fixpoint operator

The *least fixpoint operator* [3,5,6] actually does not work on relations but on queries, it transforms them into other ones. We first point out for which queries we shall consider the least fixpoint operator

Definition 4 1
An lfp expression is a unary scheme preserving expression $E(x)$ such that for all relations r and s for which $E(r)$ and $E(s)$ are defined
1 $r \subseteq E(r)$ *(increasing)*,
2 $r \subseteq s$ *implies* $E(r) \subseteq E(s)$ *(monotone)*

If $E(x)$ is an lfp expression and r is a relation, then $E^*(r)$ is defined if and only if $E(r)$ is defined and must

in that case be interpreted [20] as the smallest relation s containing r for which $E(s) = s$. A straightforward argument shows that $E^*(r) = \bigcup_{i=1}^{\infty} E^i(r)$. Note that for each relation r for which $E(r)$ is defined, $E^*(r)$ can always be computed, since for some positive integer k, $E^k(r) = E^{k+1}(r) = E^{k+2}(r) =$

We now formally define the lfp closure of the nested relational algebra (respectively the powerset algebra)

Definition 4 2
The lfp closure of the nested algebra $\mathcal{N}$ (respectively the powerset algebra $\mathcal{P}$) is the smallest set $\mathcal{N}^*$ (respectively $\mathcal{P}^*$) satisfying
1 $\mathcal{N}^*$ *(respectively $\mathcal{P}^*$) contains all primitive expressions,*
2 $\mathcal{N}^*$ *(respectively $\mathcal{P}^*$) is closed under the basic operators of the nested algebra (respectively the powerset algebra),*
3 *For each lfp expression $E(x)$ in $\mathcal{N}^*$ (respectively $\mathcal{P}^*$), $E^*(x)$ is also in $\mathcal{N}^*$ (respectively $\mathcal{P}^*$)*

Example 4 1
A classical example of a query that can be constructed from a flat relational algebra query using the lfp operator, is the *transitive closure* of a binary flat relation. We show that this query can be expressed in the lfp closure of the nested algebra. Let $r = (\{A, B\}, \omega)$. Consider the following lfp expression

$$E(x) = x \cup \pi_{\{A,B\}} \sigma_{C=D}(\rho_{C \leftarrow B}(x) \times \rho_{D \leftarrow A}(x))$$

Obviously the transitive closure of r equals $E^*(r)$

As the transitive closure of a binary relation cannot even be computed in the nested relational algebra [18], it follows that the *lfp closure of the nested algebra* is strictly more expressive than the ordinary nested algebra. Actually, it has been shown in [10] (and independently also in [2]) that the lfp closure of the nested algebra is equivalent to the powerset algebra

Theorem 4 1
The lfp closure of the nested relational algebra, the powerset algebra and the lfp closure of the powerset algebra have the same expressive power

In the following two sections, we propose some other possible extensions of the nested relational algebra and show their equivalence to the powerset algebra. Because of space limitations, most of the proofs are either omitted or replaced by a sketch. Interested readers can find more details in [11]

5. Introducing while-loops in the algebra

In [4], Chandra and Harel introduce the language QL to express queries on flat relations. QL basically consists of the classical flat relational algebra, augmented

228

with two very powerful features *unranked variables*
(1 e not associated with a fixed scheme) and a *while-
construct* Both features give to QL the computing
power of Turing Machines and hence all computable
queries on flat relations can be expressed in QL

Of course, we should not hope that e g the power-
set algebra would have the same expressive power as
QL, since powerset algebra expressions have a fixed
scheme In [5] however, Chandra and Harel introduced
and studied the language RQL which is syntactically
almost identical to QL, but with the restriction that all
variables are ranked (1 e associated to a fixed scheme)
Therefore, we wanted to see what happens to the ex-
pressive power of the nested algebra, when it is aug-
mented with a while-construct We shall show in this
section that this augmentation results in exactly the
expressiveness of the powerset algebra, provided slight
extensions are made to deal with undefinedness

5.1. The v-operator

In Definition 2 2 2, we defined expressions in such a
way that their result for some finite sequence of rela-
tions could never be undefined, "illegal" expressions or
"illegal" substitutions were simply not allowed The
situation for while-loops however is clearly different,
although not yet formally introduced, one can read-
ily see that the result of a while-loop can be unde-
fined in case it runs indefinitely Moreover, the result
of a while-loop being undefined can really depend on
the actual instances used to start up the while-loop
Therefore, we shall extend the nested algebra (respec-
tively the powerset algebra) slightly to deal with this
situation

First, for each composed attribute $X \in \mathcal{U} - U$, we
add to $\mathcal{I}_X$, the set of all relation instances over X,
the new value $?_X$, which should be interpreted as the
"undefined instance" over the scheme X Note that,
by this convention, the values $?_X$ *cannot* appear on
a lower level in a nested relation Of course, we now
must point out how the basic operators of the nested
algebra (respectively the powerset algebra) work on
this new values Very straightforwardly, we agree that
whenever a basic operator works on a relation (having
the appropriate scheme) with an "undefined instance",
then the result is also a relation with an "undefined
instance", the scheme of which is defined by the oper-
ation

Example 5 1 1
Let (Ω, ω) be a relation Let $\Omega' \in \mathcal{U} - U$ have no
elementary attributes in common with Ω Assume
furthermore that $X \subseteq \Omega$ Then
- $(\Omega, ?_\Omega) \cup (\Omega, \omega) = (\Omega, ?_\Omega)$,
- $(\Omega, \omega) \times (\Omega', ?_{\Omega'}) = (\Omega \cup \Omega', ?_{\Omega \cup \Omega'})$,
- $\nu_X(\Omega, ?_\Omega) = ((\Omega - X) \cup \{X\}, ?_{(\Omega-X) \cup \{X\}})$ ∎

Obviously, we need an operator that can generate an
"undefined instance" in an input-dependent way

Definition 5 1 1
Let (Ω, ω) be a relation The v-operator is defined by

$$v(\Omega, \omega) = \begin{cases} (\Omega, \omega) & \text{if } \omega \neq \emptyset \\ ?_\Omega & \text{if } \omega = \emptyset \end{cases}$$
$$v(\Omega, ?_\Omega) = ?_\Omega$$

∎

In words, the v-operator (where "v" stands for "unde-
fined") maps each non-empty relation to itself and the
empty relation to that newly introduced value which
corresponds to the scheme of the relation

We now consider the *extended nested algebra* (respec-
tively the *extended powerset algebra*) whose set of ex-
pressions is defined by the basic operators of the nested
algebra (respectively the powerset algebra), augment-
ed with the v-operator These generalized expres-
sions are called *extended naes (enaes)* (respectively
extended paes (epaes))

Expressions in which the v-operator occurs are in gen-
eral rather difficult to handle Fortunately, it will turn
out that we only have to consider a subclass of all
enaes (respectively epaes) to generate the extended
nested algebra (respectively the extended powerset al-
gebra) In preparation of this normalization, we first
define the *global unnesting* and the *global nesting* op-
erator

Definition 5 1 2
- *Let $(\{\Omega\}, \omega)$ be a relation The global unnesting
  $M(\{\Omega\}, \omega)$ equals $\mu_\Omega(\{\Omega\}, \omega)$,*
- *Let (Ω, ω) be a relation Then the global nesting
  $N(\Omega, \omega)$ equals*

$$N(\Omega, \omega) = \begin{cases} (\{\Omega\}, ?_{\{\Omega\}}) & \text{if } \omega = ?_\Omega \\ (\{\Omega\}, \{\omega\}) & \text{otherwise} \end{cases}$$

∎

Lemma 5 1 1
*There exists naes (and hence enaes) that express the
global unnesting and the global nesting* ∎

Since the global nest operator never returns an emtpy
relation, even if the relation on which it is applied
is empty, it can be used to "shield" relations from
unwanted side-effects of the v-operator In this way, it
is possible to pull back an occurrence of the v-operator
to the end of the expression it appears in Finally,
global unnesting can then be used to undo the global
nesting These ideas motivate the following definition

Definition 5 1 3
*A normalized enae (respectively epae) is an enae (re-
spectively epae) $MvE(x, y,)$, where $E(x, y,)$ is a
nae (respectively pae)* ∎

Lemma 5 1 2

The set of all normalized enaes (respectively epaes) has the same expressive power as the extended nested algebra (respectively the powerset algebra)

Proof By straightforward induction ∎

5.2. While-loop queries

In this subsection, we shall introduce while-loops in the algebra Therefore we first define in a formal way what we understand by a *while-loop query*

Definition 5 2 1

Let $r_1 = (\Omega_1, \omega_1)$ and $r_2 = (\Omega_2, \omega_2)$ be relations Let $E_1(y)$ be a unary expression, defined on relations with scheme Ω_2, and let $E_2(x, y)$ and $E_3(x, y)$ be binary expressions, defined on pairs of relations over the schemes Ω_1 and Ω_2 respectively, for which Ω_1 respectively Ω_2 are the schemes of the resulting relations A while-loop query of type (Ω_1, Ω_2) is a binary query Q for which $Q(r_1, r_2)$ can be written as the result of a program of the following form

$$
\begin{aligned}
&\textbf{begin} \\
&\quad r = r_1, \\
&\quad s = r_2, \\
&\quad \textbf{while } E_1(s) \neq (\Omega^{E_1}, \emptyset) \\
&\quad\quad \textbf{do} \\
&\quad\quad\quad r = E_2(r, s), \\
&\quad\quad\quad s = E_3(r, s) \\
&\quad\quad \textbf{od}, \\
&\quad Q(r_1, r_2) = r \\
&\textbf{end}
\end{aligned}
$$

We assume that $Q(r_1, r_2)$ takes the value $(\Omega_1, {}^?\Omega_1)$ if the while-loop runs indefinitely ∎

In the while-loop query above, the actual computation is done by the expression $E_2(x, y)$, $E_3(x, y)$ is used to compute the test relation whereas $E_1(y)$ expresses the test condition In Definition 5 2 1, we chose a test of the form $E(s) \neq (\Omega^E, \emptyset)$ One might also have chosen $E(s) \neq E'(s)$, $E(s) = E'(s)$, $E(s) = (\Omega^E, \emptyset)$, $E(s) = (\Omega^E, \{\emptyset\})$, $E(s) \neq (\Omega^E, \{\emptyset\})$, $s \neq \emptyset$, $s = \emptyset$, We leave it to the reader to convince himself that all these choices are equivalent with respect to expressive power

Note that, since, apart from possibly "undefined instances", no new values are introduced during the execution of the while-loop above, it is decidable whether or not a while-loop will end

If the expressions in Definition 5 2 1 are all enaes (respectively epaes), we say that Q is a *while-loop query*

in the extended nested algebra (respectively the extended powerset algebra) By starting from enaes (respectively epaes) and by recursively associating a binary expression to each while-loop query, one can define the *while-closure of the extended nested algebra* (respectively *the extended powerset algebra*)

Obviously, the while-closure of the extended nested algebra is more powerful than the extended nested algebra itself

Example 5 2 1

We show that the transitive closure of a binary flat relation can be expressed in the while-loop closure of the nested algebra Let $E(x)$ be the expression defined in Example 4 1 Now consider the following while-loop query of type $(\{A, B\}, \{A, B\})$

$$
\begin{aligned}
&\textbf{begin} \\
&\quad r = r_1, \\
&\quad s = r_2, \\
&\quad \textbf{while } E(s) - s \neq (\{A, B\}, \emptyset) \\
&\quad\quad \textbf{do} \\
&\quad\quad\quad r = E(r), \\
&\quad\quad\quad s = E(s) \\
&\quad\quad \textbf{od}, \\
&\quad Q(r_1, r_2) = r \\
&\textbf{end}
\end{aligned}
$$

Obviously, the transitive closure of a relation $r_1 = (\{A, B\}, \omega)$ can be expressed as $Q(r_1, r_1)$ ∎

5.3. The expressiveness of the while-construct

In this subsection, we show that the while-closure of the extended nested relational algebra, the powerset algebra and the while-closure of the extended powerset algebra have the same expressive power

Theorem 5 3 1

The while-closure of the extended nested algebra, the extended powerset algebra and the while-closure of the extended powerset algebra have the same expressive power

Proof We only give a sketch The theorem will be proven if we can demonstrate

1 The powerset operator can be expressed by a while-loop query in the extended nested algebra,

2 Each while-loop query in the extended powerset algebra can be expressed in the extended powerset algebra

The first item is the easier to show Let $r_1 = (\Omega_1, \omega_1)$ be an arbitrary relation First, we mention that it is possible to construct an nae $E_1(x)$ such that $E_1(r_1)$ is a relation with scheme $\{\Omega_1\}$ that consists of all subsets of ω of size at most 1 Suppose now that we are given a relation $r = (\{\Omega_1\}, W)$ It is also possible to construct an nae $E_2(y)$ such that $E_2(r) = (\{\Omega\}, \{w_1 \cup w_2 \mid$

230

$w_1, w_2 \in W\}$ Hence, if r consists of all subsets of r_1 up to size i, then $E_2(r)$ consists of all subsets of r up to size $2i$ It is now straightforward to construct a while-loop query Q such that $\Pi(r_1) = Q(E_1(r_1), E_1(r_1))$

The second part of the proof is much trickier We only give the main ideas here First of all, it is important that we may assume without loss of generality that all the expressions in a while-loop query are normalized Finally, by subtilly using global nesting, the powerset operator and the least fixpoint operator, we can define an expression that constructs the set of all intermediate results of the while-loop query Then a selection according to the condition of the while-loop will yield the correct result, if no relation in the aforementioned set satisfies the condition, an application of the v-operator will yield an "undefined" relation as output of the while-loop query ∎

5.4. The if-then-else construct

We might also want to introduce an if-then-else construct in the extended nested algebra (respectively the extended powerset algebra)

<u>Definition 5 4 1</u>

Let $r_1 = (\Omega_1, \omega_1)$ and $r_2 = (\Omega_2, \omega_2)$ be relations Let $E_1(y)$ be a unary expression, defined on relations with scheme Ω_2, and let $E_2(x)$ and $E_3(x)$ be unary expressions, defined on relations with scheme Ω_1 such that $\Omega^{E_2} = \Omega^{E_3}$ An if-then-else-query of type (Ω_1, Ω_2) is a binary query Q for which $Q(r_1, r_2)$ can be written as the result of a program of the following form

> **begin**
>   **if** $E_1(r_2) \neq (\Omega^{E_1}, \emptyset)$
>   **then**
> $Q(r_1, r_2) = E_2(r_1)$
>   **else**
> $Q(r_1, r_2) = E_3(r_1)$
> **end**

∎

We have [9]

<u>Theorem 5 4 1</u>

If-then-else queries in the extended nested algebra (respectively the extended powerset algebra) can be expressed in the extended nested algebra (respectively the extended powerset algebra) ∎

6. Introducing for-loops in the algebra

In the beginning of the previous section, we discussed the language QL introduced in [4] Due to the fact that variables are not ranked, i e that the scheme of

the relation they represent can grow wider during the computation process, it is possible to simulate counting in QL, which, in combination with the presence of while-loops gives QL the power of general Turing Machines Of course we cannot expect the same for the powerset algebra, where all expressions are ranked However, we still retain some of that, since in this section we shall show that a certain type of for-loop can be expressed in the powerset algebra

<u>Definition 6 1</u>

Let $r_1 = (\Omega_1, \omega_1)$ and $r_2 = (\Omega_2, \omega_2)$ be relations Let $E_1(x)$ be a unary scheme preserving expression, defined on relations with scheme Ω_1 A for-loop query of type (Ω_1, Ω_2) is a binary query Q for which $Q(r_1, r_2)$ can be written as the result of a program of the following form

> **begin**
> $r = r_1,$
>   **for** $i = 1$ **to** $|r_2|$
>   **do**
> $r = E_1(r),$
>   **od,**
> $Q(r_1, r_2) = r$
> **end**

In this program $|r_2|$ stands for $|\omega_2|$, i e the number of tuples of r_2 We assume that $Q(r_1, r_2)$ takes the value $(\Omega_1, {}^?\Omega_1)$ if $\omega_2 = {}^?\Omega_2$ ∎

<u>Example 6 1</u>

Again, let us have a look to the transitive closure of a binary flat relation Let $E(x)$ be the expression defined in Example 4 1 Consider the following for-loop query of type $(\{A, B\}, \{A, B\})$

> **begin**
> $r = r_1,$
>   **for** $i = 1$ **to** $|r_2|$
>   **do**
> $r = E(r),$
>   **od,**
> $Q(r_1, r_2) = r$
> **end**

Now let $E'(x) = \pi_{\{A\}}(x) \times \pi_{\{B\}}(x)$ Obviously, the transitive closure of a relation $r_1 = (\{A, B\}, \omega)$ can be expressed as $Q(r_1, E'(r_1))$ ∎

As in the previous section, we can define the *for-closure of the extended nested algebra* (respectively *the extended powerset algebra*) We are now going to show that both closures are equivalent to the extended powerset algebra

<u>Theorem 6 1</u>

The for-closure of the extended nested algebra, the extended powerset algebra and the for-closure of the extended powerset algebra have the same expressive power

<u>Proof</u> Again, the easiest direction is showing that the powerset operator can be expressed using a for-loop query in the extended nested relational algebra Let $r_1 = (\Omega_1, \omega_1)$ Reconsider the proof of Theorem 5 3 1 Since the while-loop constructed in the first part of that proof needs to cycle at most $|r_1|$ times (this is a very rough estimate) to compute the desired result, it is straightforward to transform that while-loop into a for-loop

Now let $r_1 = (\Omega_1, \omega_1)$ and $r_2 = (\Omega_2, \omega_2)$ be relations and reconsider the for-loop query Q of Definition 5 1 in which the expressions are supposed to be epaes We show that this query can be expressed in the powerset algebra, or, equivalently, in the while-loop closure of the powerset algebra First, let $r = (\{\Omega_2\}, \mathcal{W})$ There exists an enae $E_2(y)$ such that $E_2(r)$ is a relation with scheme $\{\Omega_2\}$ consisting of those elements of $\mathcal{W}$ that are *not* minimal with respect to inclusion In particular, it follows that $|r_2|$ is the smallest integer $\imath$ for which $E_2^\imath(\Pi(r_2) - (\{\Omega_2\}, \{\emptyset\})) = \emptyset$ It is now straightforward to construct a while-loop query Q' such that $Q'(r_1, r_2) = Q(r_1, \Pi(r_2) - (\{\Omega_2\}, \{\emptyset\}))$ ∎

Acknowledgment

References

[1] S Abiteboul, C Beeri, "On the Power of Languages for the Manipulation of Complex Objects" *draft*, April 1987

[2] S Abiteboul, C Beeri, *personal communications*

[3] A V Aho, J D Ullman, "Universality of Data Retrieval Languages", *Proc 6th POPL*, San-Antonio, 1979, pp 110–117

[4] A K Chandra, D Harel, "Computable Queries for Relational Data Bases", *Journal of Computer and System Sciences 21*, 1980, pp 156–178

[5] A K Chandra, D Harel, "Structure and Complexity of Relational Queries", *Journal of Systems and Computers Sciences 25*, 1985, pp 99–128

[6] A K Chandra, D Harel, "Horn Clause Queries and Generalizations", *Journal of Logic Programming 1*, 1985, pp 1–15

[7] E F Codd, "A Relational Model of Data for Large Shared Data Banks", *Communications of ACM 13 6*, June 1970, pp 377–387

[8] M Gyssens, "The extended nested relational algebra", *Techn Rep 87-11*, Univ of Antwerp, 1987

[9] M Gyssens, G Houben, J Paredaens, D Tahon, *unpublished results*

[10] M Gyssens, D Van Gucht, "The Powerset Operator as an Algebraic Tool for Understanding Least Fixpoint Semantics in the Context of Nested Relations", *Techn Rep 233*, Indiana Univ, Bloomington, 1987

[11] M Gyssens, D Van Gucht, "The Powerset Algebra as a Result of Adding Programming Constructs to the Nested Relational Algebra", *Techn Rep 87-26*, Univ of Antwerp, 1987

[12] G Houben, J Paredaens, D Tahon, *personal communications*

[13] R Hull, "On the Expressive Power of Database Queries with Intermediate Types", *Techn Rep*, Univ of South California, Los Angeles, 1987

[14] G Jaeschke, H J Schek, "Remarks on the Algebra on Non First Normal Form Relations", *Proc 1st PODS*, Los Angeles, 1982, pp 124–138

[15] G M Kuper, M Y Vardi, "A New Approach to Database Logic", *Proc 3rd PODS*, Waterloo, 1984, pp. 86–96

[16] A Makinouchi, "A Consideration of Normal Form of Not-Necessarily-Normalized Relations in the Relational Data Model", *Proc 3rd VLDB*, Tokyo, 1977, pp 447–453

[17] G Ozsoyoğlu, Z M Ozsoyoğlu, V Matos, "Extending Relational Algebra and Relational Calculus with Set-Valued Attributes and Aggregate Functions", *ACM Transactions On Database Systems 12 4*, Dec 1987, pp 566–592

[18] J Paredaens, D Van Gucht, "Possibilities and Limitations of Using Flat Operators in Nested Algebra Expressions", *Proc 7th PODS*, Austin, 1988

[19] M A Roth, H F Korth, A Silberschatz, "Theory of Non-First-Normal-Form Relational Databases", *Techn Rep TR-84-36*, Univ of Texas, Austin, 1986

[20] A Tarski, "A Lattice-Theoretical Fixpoint Theorem and its Applications", *Pacific Journ of Mathematics 5*, 1955, pp 285–309

[21] S J Thomas, P C Fischer, "Nested Relational Structures", *The Theory of Databases*, P C Kanellakis, ed , JAI Press, 1986, pp 269–307

[22] D Van Gucht, "On the Expressive Power of the Extended Relational Algebra for the Unnormalized Relational Model", *Proc 6th PODS*, San Diego, 1987, pp 302–312