PUC

ISSN 0103-9741

Monografias em Ciência da Computação
nº 04/93

Dealing with Non-Constructive Specifications Involving Quantifiers

Armando M. Haeberer
Gabriel A. Baum
Gunther Schmidt

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

PUC RIO - DEPARTAMENTO DE INFORMÁTICA ISSN 0103-9741

Monografias em Ciência da Computação, Nº 04/93
Editor: Carlos J. P. Lucena May, 1993

Dealing with Non-Constructive Specifications Involving Quantifiers*

Armando M. Haeberer
Gabriel A. Baum**
Gunther Schmidt***

* This work has been sponsored by the Secretaria de Ciência e Tecnologia da Presidência da República Federativa do Brasil.

** Universidad Nacional de La Plata, Faculdad de Ciencias Exatas, Departamento de Informática, 1 y 50, La Plata, Buenos Aires, Argentina

*** Univesität der Bundeswehr, München, D-8014 Neubiberg, Alemanha

In charge of publications:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC Rio — Departamento de Informática
Rua Marquês de São Vicente, 225 — Gávea
22453-900 — Rio de Janeiro, RJ
Brasil
Tel. +55-21-529 9386 Telex +55-21-31048 Fax +55-21-511 5645
E-mail: rosane@inf.puc-rio.br
techrep@inf.puc-rio.br (for publications only)

Dealing with Non-Constuctive Specifications Involving Quantifiers

Armando M. Haeberer ♠
Gabriel A. Baum ♣
Gunther Schmidt ⊛

armando@inf.puc-rio.br

ABSTRACT

The work presented here has its focus on the formal construction of programs out of non-constructive specifications involving quantifiers. This is accomplished by means of an extended abstract algebra of relations whose expressive power is shown to encompass that of first-order logic. Our extension is the first finitary one that solves the classic issue of lack of expressiveness of abstract relational algebras first stated by Tarski and later formally treated by Maddux, Németi, etc. First we compare our extension with classic approaches to expressiveness and our axiomatization with modern approaches to products. Then, we introduce some non-fundamental operations. One of them, the *relational implication*, is shown to have heavy heuristic significance both in the statement of Galois connections for expressing relational counterparts for universally quantified sentences and for dealing with them. In the last sections we present two smooth program derivations based on the theoretical framework introduced previously.

Keywords: Formal program construction, Relational algebras, Algebraiztion of logic.

RESUMO

O trabalho aqui apresentado focaliza a construção formal de programas a partir de especificações não-construtivas contendo quantificadores. Isto é obtido através de uma álgebra abstrata de relações, cujo poder expressivo demontra-se que abrange o da lógica de primeira ordem. Nossa extensão é a primeira finitária que resolve o problema clássico da falta de expressividade das álgebras relacionais abstratas. Tal problema é aquele enunciado originalmente por Tarski e mais tarde tratado formalmente por Maddux, Németi, etc. Nas primeiras seções, comparamos nossa extensão com abordagens clássicas e nossa axiomatização com abordagens modernas para produtos. A seguir, introduzimos algumas operações não fundamentais. Mostramos que a operação chamada de implicação relacional. tem um forte significado heurístico, tanto no enunciado de conexões de Galois para contrapartidas relacionais de sentenças quantificadas universalmente quanto para seu tratamento posterior. Nas últimas seções, apresentamos duas derivações completamente sintáticas baseadas no arcabouço teórico introduzido previamente.

Palavras Chave: Construção formal de programas, Álgebras relacionais, Algebrização da lógica.

♠ Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). Departamento de Informática, Av. Marquês de São Vicente 225, 22453 Rio de Janeiro, RJ, Brazil.

♣ Universidad Nacional de La Plata, Facultad de Ciencias Exactas, Departamento de Informática, 1 y 50 La Plata, Buenos Aires, Argentina.

⊛ Universität der Bundeswehr, München, D-8014 Neubiberg.

1 Introduction

The last few years have witnessed a renewed interest of the computing science community in relational programming calculi. The main reason for such interest is the postulated advantage of such calculi for dealing with non-determinism. Also, the claim about its fitness for expressing what-to-do instead of how-to-do specifications should not be ignored.

In the meantime, another major issue of whether or not a programming calculus should encompass the whole path from non-constructive specifications to programs has been almost completely neglected. In the case of some methods, as CIP, such disregard simply leads to a bias in the research work favouring algorithm optimisation to the detriment of algorithm construction. However, looking at program calculi in general, and at relational ones in particular, the situation is worse because any decision about this issue biases harmfully the choice of an underlying theoretical framework. In the particular case of relational programming calculi, the question is as crucial as whether or not Abstract Relational Algebras qualify as an appropriate basis for them.

Along this paper we first argue, in Section 2, that, under the light of the restricted expressiveness of Abstract Relational Algebras, initially recognised by Tarski [Tar41] and later on formally stated by Henkin and Monk [Hen74] as well as by Maddux and Németi [Mad83, 91, Ném91], we should abandon them as a basis of relational programming calculi. We also examine the implications of the so-called representability problem over programming calculi development. In the following section, we first analyse a well-known solution to representability and subsequently some classical solutions to the expressiveness problem, namely, Cylindric and Polyadic algebras. Later on, in Section 4 , we propose a formal extension to Abstract Relational Algebras, prove that it has the same expressive power than first-order logic with equality, discuss its representability and compare it with both classical algebras for first-order logic and other extensions to Abstract Relational Algebra under use, for supporting relational programming calculi.

In the second part of the paper, i.e., in Section 5 , we first introduce in Subsection 5.1 some results about partial identities as a tool for dealing with types within an homogeneous framework as well as a refinement order relation among relations. In the following subsection we introduce an example of the calculation of a recursive expression out of a first-order non-constructive specification involving an existential quantifier. In Subsection 5.4 we recall the concept of residuals and introduce a convenient heuristic way for considering them. Finally, in Subsection 5.5 we illustrate with a detailed example how to derive recursive expressions out of non-constructive specifications involving universal quantifiers[1].

---

1 The properties of the "classical" relational operations are well-known and can be found elsewhere [Sch89].

2 Are Abstract Relational Algebras an Appropriate Support for Programming Calculi?

One of the various presentations (axiomatizations) for Abstract Relational Algebras is that due to L. Chin and A. Tarski [Chi51]. The version we choose here is that presented by G. Schmidt and T. Ströhlein [Sch85a][2].

Definition 1 An *Abstract Relational Algebra* is a structure $A = \langle \mathfrak{R}, +, \bullet, \bar{}, ;, ^{\mathsf{T}} \rangle$ over a non-empty set $\mathfrak{R}$, such that .

(1) $\langle \mathfrak{R}, +, \bullet, \bar{} \rangle$ is a complete atomic Boolean Algebra. Its *zero element* will be denoted by 0, and its *unit element* by ∞. The symbol $\subset$ is used for the ordering with respect to the lattice structure and is called *inclusion*

(2) $\langle \mathfrak{R}, ; \rangle$ is a semigroup with exactly one *identity element* which is denoted by 1, i.e. $(r;s);t = r;(s;t)$ & $r;1 = 1;r = r$

(3) $r;s \subset t \leftrightarrow r^{\mathsf{T}};\bar{t} \subset \bar{s} \leftrightarrow \bar{t};s^{\mathsf{T}} \subset \bar{r}$ *Schröder rule*

(4) $r \neq 0 \rightarrow \infty;r;\infty = \infty$ *Tarski rule*

The standard models for Abstract Relational Algebras so defined are called Proper Relational Algebras. A Proper Relational Algebra [Tar41] is a first-order theory satisfying the following extralogical axioms,

i.	$\forall x \forall y\,(x \infty y)$	*vii.*	$\forall x \forall y\,(x r^{\mathsf{T}} y \leftrightarrow y r x)$
ii.	$\forall x \forall y\,(\neg x 0 y)$	*viii.*	$\forall x \forall y\,(x r + s\, y \leftrightarrow x r y \vee x s y)$
iii.	$\forall x\,(x 1 x))$	*ix.*	$\forall x \forall y\,(x r \bullet s\, y \leftrightarrow x r y \wedge x s y)$
iv.	$\forall x \forall y \forall z\,((x r y \wedge y 1 z) \rightarrow x r z)$	*x.*	$\forall x \forall y\,(x r \dagger s\, y \leftrightarrow (\forall z)(x r z \vee z s y)$
v.	$\forall x \forall y\,(x \wp y \leftrightarrow \neg x 1 y)$	*xi.*	$\forall x \forall y\,(x r ; s\, y \leftrightarrow (\exists z)(x r z \wedge z s y)$
vi.	$\forall x \forall y\,(x \bar{r}\, y \leftrightarrow \neg x r y)$	*xii.*	$r = s \leftrightarrow \forall x \forall y\,(x r y \leftrightarrow x s y)$

This is a two-sorted theory, i.e., its variables are of two kinds, one ranging over *individuals* (denoted here by x, y, z, . . .) while the other ranges over relations (denoted here by italic letters r, s, t, ...). The atomic sentences are of the form $x r y$ (with intended meaning "x is in relation r with y") and $r = s$ (where the symbol = denotes equality over relations). Variables on individuals range over a fixed set A, while variables on relations range over some subsets of $A \times A$ –this *simplicity* is introduced by the inclusion of the so-called *Tarski rule* within the axiomatization of Abstract Relational Algebras.

The symbols introduced by Tarski are; in our notation, ∞ (for the universal relation), 0 (for the null relation), 1 (for the identity relation) and $\wp$ (for the diversity relation), as relational constants, together with the following operations on relations $\bar{}$ (complement), $^{\mathsf{T}}$ (converse), + (union), • (intersection), † (relative addition), and ; (relative product).

Tarski [Tar41] posed two questions that are not only fundamental for the fields of Relational Algebras and Algebraic Logic but that we consider of primordial importance for relational programming calculi.

---

2 For an up-to-date history of the development of Relational Algebras and a deep study of its relation with first-order logic see [Mad91].

- Can every property of relations, relations among relations, etc., that can be defined in a Proper Relational Algebra[3], be expressed in an Abstract Relational Algebra?
- Is every model of (1) to (4) of Definition 1 isomorphic to a Proper Relational Algebra (its standard model)? In other words, is every Abstract Relational Algebra representable?

It has been known for many years that the answer to both questions is negative.
The first one was answered by Tarski himself by showing two simple first-order expressions

$$(\forall x)(\forall y)(\forall z)(\exists u)(x\,r\,u \wedge y\,r\,u \wedge z\,r\,u) \qquad (5)$$

$$(\exists x)(\exists y)(\exists z)(\exists u)(x\,r\,y \wedge x\,r\,z \wedge x\,r\,u \wedge y\,r\,z \wedge y\,r\,u \wedge z\,r\,u) \qquad (6)$$

that cannot be expressed within his relational calculus (abstract relational algebra). For instance, no sentence of the relational calculus is satisfied by exactly the same relations that satisfy expression (5). Some years later Henkin and Monk [Hen74] and Maddux [Mad83, Mad91] stated that Relational Algebras correspond to predicate logic in which only three individual variables are allowed and all predicates are at most binary.
The *Representation Problem* (i.e., whether every abstract relational algebra is representable) received a negative answer from Roger Lyndon [Lyn50]. Lyndon constructed a finite nontrivial simple relational algebra which is non-representable.
Our view of Programming Calculi is, on one hand, that such calculi should be unifying formalisms for writing either non-constructive or constructive specifications and for calculating programs out of them. On the other hand, we claim that *actual formal programming methodologies* will contain, together with specification construction and validation tools, not one but various interacting programming calculi (then, the possibility of inter-translation of terms between them is a must). Thus, from our viewpoint, we should provide not only programming calculi exhibiting –at least[4]– the expressive power of first-order logic with equality, but also calculi for translating back and forth first-order expressions to terms of such programming calculi.
Moreover, if we do not want to fall into a *linguistic trap* resulting in divorce from reality (i.e., practical software engineering) as had happened in the case of logical empiricists with respect to the explication of the construction of scientific theories, we should bear in mind that while calculating a program we can often interpret terms as input-output (semantical) relations.

---

3 Tarski used the terms *Elementary Theory of Binary Relations* and *Calculus of Binary Relations* for Proper this Relational Algebra and Abstract Relational Algebra, respectively.

4 If we want to talk about termination within the programming calculus itself, we should require that such calculi have an expressive power which encompasses that of first-order logic.

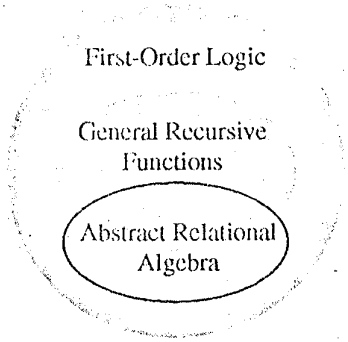

Fig. 1. Relative expressiveness of Abstract Relational Algebra

So, if we consider,

- Tarski's statement that the answer to the first question remains negative even if we enrich Abstract Relational Algebra by the addition of any finite number of new constants denoting fixed relations, properties of relations, operations on relations, and so on, provided that all concepts introduced in this way are invariant under any one-to-one mapping of the class of individuals on itself, as well as,
- the non representability of Abstract Relational Algebras,

we should abandon Abstract Relational Algebras as the basis for programming calculi, especially if we consider that the actual expressiveness situation is that depicted in figure 1.

3 "Classical" Solutions to Representability and Expressiveness Problems

3.1 Point Axiom and the Solution of the Representability Problem

The solution to the problem of the non-representability of Abstract Relational Algebras is well known, requiring the inclusion of an extra axiom. For example G. Schmidt and T. Ströhlein have proposed [Sch85a] the introduction of an axiom that they have called *point axiom*. Let us analyse succinctly their proposal.

Definition 2 Let x be a relation. We call

(7) x a *vector* iff $x = x;\infty$

(8) x a *point* iff x is a vector, $x \neq 0$, $x;x^{\mathsf{T}} \subset 1$

Notice that the interpretation of a vector in a Proper Relational Algebra is a relation $B \times A \subset A \times A$. Then, since A, the fixed set we have taken in the standard model we are dealing with, is invariant for every vector, such vector is one of the ways[5] of representing sets within an Algebra of Relations. In fact this is the way chosen by

---

5 For a detailed treatment of how to represent and *internalise* sets within a Relational Algebra see [Vel92] .

Hoare and He [Hoa86a, 86b]. Along the same line of thinking, notice that points are then the representation of unary sets.
Continuing with the representation problem we will say that an Abstract Relational Algebra satisfies the *point axiom* if $r \neq 0 \rightarrow (\exists x)(\exists y)\left(x, y \text{ points} \wedge x\,;\, y^{\mathsf{T}} \subset r\right)$. Now we can state the main representability result obtained by G. Schmidt and T. Ströhlein, i.e.,

Theorem 3 An Abstract Relational Algebra A satisfying the point axiom is a Proper Relational Algebra. More precisely A is representable.

The reader interested in the proof of this theorem as well as in the detailed treatment of this issue is referred to [Sch85a].

3.2 Classical Algebras for First-order Logic
Since Boolean Algebra has the expressive power of propositional logic, it seems natural to try to obtain an algebra for first-order logic (in particular for first-order logic with equality) by extending Boolean Algebra with some operations so as to cope with quantifiers (and the equality predicate).

Boolean Algebras with Operators. B. Jónsson and A. Tarski defined the concept of *Boolean Algebras with Operators* [Jón51], i.e., an algebra resulting of extending a Boolean Algebra with a number, not necessarily finite, of operations satisfying certain conditions. They also show [Jón52] that Abstract Relational Algebra is a Boolean Algebra with Operators – unfortunately with not enough expressive power to become an algebra of first-order logic.
Speaking formally [Jón51], by an *algebra* we shall mean a finite, infinite, or transfinite sequence,

$$\mathfrak{G} = \left\langle \mathfrak{A}, f_0, f_1, \cdots, f_\xi, \cdots \right\rangle$$

where $\mathfrak{A}$ is a non-empty set and each f_ξ is a function of some finite rank m_ξ with domain $\mathfrak{A}^{m_\xi}$ and range a subset of $\mathfrak{A}$. In this context the term *operation* is sometimes used instead of the term *function* ; $f_0, f_1, \ldots, f_\xi, \ldots$ are referred as *fundamental operations* of $\mathfrak{A}$.
Consider now the Boolean Algebra $\mathfrak{B} = \langle \mathfrak{A}, +, 0, \bullet, \infty \rangle$, then,

Definition 4 A function f on $\mathfrak{A}^m$ to $\mathfrak{A}$ is called

(i) *normal* if, given any j<m and a sequence $x \in \mathfrak{A}^m$ such that $x_j = 0$, we always have $f(x) = 0$;
(ii) *monotonic* if, given two sequences $x, y \in \mathfrak{A}^m$ such that $x \leq y$, we always have $f(x) \leq f(y)$;
(iii) *additive* if, given any j≤m and two sequences $x, y \in \mathfrak{A}^m$ such that $x_p = y_p$ whenever j≠p≤m, we always have $f(x+y) = f(x) + f(y)$;
(iv) *completely additive* if, given any j≤m, a non-empty set I, and sequences $x^{(i)} \in \mathfrak{A}^m$ with $i \in I$ such that $x_p{}^{(i)} = y_p{}^{(i')}$ whenever $i, i' \in I$ and j≠p≤m, and such that $\sum_{i \in I} x^{(i)}$ exists, then $\sum_{i \in I} f\left(x^{(i)}\right)$ exist and we have $f\left(\sum_{i \in I} x^{(i)}\right) = \sum_{i \in I} f\left(x^{(i)}\right)$.

Now we are able to introduce the following,

Definition 5 By a *Boolean Algebra with Operators* we shall mean an algebra $\mathfrak{O} = \langle \mathfrak{A}, +, 0, \bullet, \infty, f_0, f_1, \cdots, f_\xi, \cdots \rangle$, such that $\langle \mathfrak{A}, +, 0, \bullet, \infty \rangle$ is a Boolean Algebra and the functions f_ξ are additive. By an *atom* of $\mathfrak{O}$ we mean an atom of the Boolean Algebra $\langle \mathfrak{A}, +, 0, \bullet, \infty \rangle$. We say that $\mathfrak{O}$ is *atomistic* if the Boolean Algebra $\langle \mathfrak{A}, +, 0, \bullet, \infty \rangle$ is atomistic. We call $\mathfrak{O}$ *complete* if the Boolean Algebra $\langle \mathfrak{A}, +, 0, \bullet, \infty \rangle$ is complete and if each of the operations f_ξ is completely additive.

Then, it is evident that every algebra satisfying the axioms (1) to (4) in Definition 1 –i.e., every Abstract Relational Algebra–, is a complete atomic Boolean Algebra with operators, the operators being ; and $^{\mathsf{T}}$.

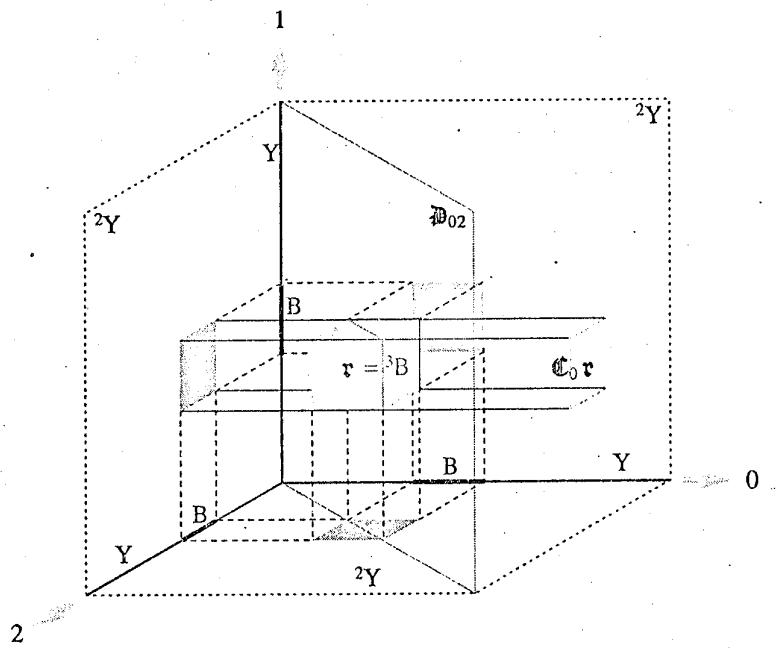

Fig. 2. $\mathbb{C}_\kappa \mathfrak{r}$ with $\kappa = 0, \alpha = 3$ and $\mathfrak{r} = \{B \times B \times B\}$

Two Classical Solutions for the Expressiveness Problem. In defining Abstract Relational Algebra, A. Tarski et al. extended Boolean Algebra with a *finite set* of operators –a way we will call *Peircean*–. The efforts of L. Chin, A. Tarski and F. B. Thompson [Chi48, Tar52] and later of Henkin, Monk and Tarski [Hal62], for defining an algebra for first-order logic with equality led to the introduction of *Cylindric Algebras*. But, in doing this, they abandon the Peircean way of extending Boolean Algebra in favour of extending it with *infinitely many operators*, thus obtaining algebras of *sets of infinitary sequences*, and likewise for Halmos's *Polyadic Algebras*. Before we begin with a succinct analysis of both algebras let us recall that the *Lindenbaun Algebra of Formulae* of a theory in a first-order language $\mathcal{L}$ is a structure

$\mathfrak{S}_{\mathcal{L}} = \langle \Phi_{\mathcal{L}}, \vee, \wedge, \neg, \mathsf{F}, \mathsf{T}, \exists v_\zeta, \iota v_\zeta v_\xi \rangle_{\zeta,\xi<\omega}$, where $\Phi_{\mathcal{L}}$ is the set of formulae of $\mathcal{L}$ and $\iota v_\zeta v_\xi$ is the equivalence class of the *equation* $v_\zeta \approx v_\xi$ for every $\zeta, \xi < \omega$. We will denote by Γ the set of sentences of $\mathcal{L}$ and by $\mathfrak{S}_{\mathcal{L}}/\equiv_\Gamma$ the quotient algebra with respect to the equivalence relation $\equiv_\Gamma$ defined as, $\varphi \equiv_\Gamma \psi$ holds iff $\varphi \leftrightarrow \psi$ is a consequence of Γ. This quotient algebra is the so-called *Tarski algebra* associated with $\mathcal{L}$ and Γ.

Cylindric Algebras. Let Y be a non-empty set. Subsets of ${}^{\alpha}Y$ may be thought of α-ary relations, and we may perform on them the usual Boolean operations $\cup, \cap$ and $-$ (the latter, complementation, being performed with respect to ${}^{\alpha}Y$). Now for each $\kappa < \alpha$ we introduce a unary operation $\mathfrak{C}_\kappa^{({}^{\alpha}Y)}$, or simply $\mathfrak{C}_\kappa$, as follows. For any $\mathfrak{r} \subset {}^{\alpha}Y$,

$$\mathfrak{C}_\kappa^{({}^{\alpha}Y)}\mathfrak{r} = \{x \in {}^{\alpha}Y\text{: there is a } y \in \mathfrak{r} \text{ with } x_\lambda = y_\lambda \text{ for all } \lambda < \alpha \text{ with } \lambda \neq \kappa\} \quad (\dagger)$$

Also we consider special sets $\mathfrak{D}_{\kappa\lambda}^{({}^{\alpha}Y)}$:

$$\mathfrak{D}_{\kappa\lambda}^{({}^{\alpha}Y)} = \{x \in {}^{\alpha}Y\text{: } x_\kappa = x_\lambda, \text{ for any } \kappa,\ \lambda < \alpha\} \quad (\ddagger)$$

Now, a *Cylindric Set Algebra of Dimension* α *with base* Y *and unit set* ${}^{\alpha}Y$, for brevity CS_α, is a structure[6]

$$\mathsf{CS}_\alpha = \left\langle \mathscr{P}({}^{\alpha}Y), \cup, \cap, -, \emptyset, {}^{\alpha}Y, \mathfrak{C}_\kappa^{({}^{\alpha}Y)}, \mathfrak{D}_{\kappa\lambda}^{({}^{\alpha}Y)} \right\rangle_{\kappa,\lambda<\alpha}$$

An idea of these notions can be easily captured by considering the case $\kappa = 0$, $\alpha = 3$ (see figure 2 with $\mathfrak{r}$ being the square relation $\mathfrak{r} = \{B \times B \times B\}$).
A subset $\mathfrak{r} \subset {}^{3}Y$ can then be pictured as a point set in three dimensional space, $\mathfrak{C}_0\mathfrak{r}$ being the cylinder generated by moving $\mathfrak{r}$ parallel to the 0-axis (this is the reason why operations $\mathfrak{C}_\kappa$ are called *cylindrifications*). $\mathfrak{D}_{02}$ consists of all points equidistant from the 0 and 2 axis, and is thus a diagonal plane (this is the reason why elements $\mathfrak{D}_{\kappa\lambda}$ are called *diagonal elements*).
The connection of CS_α's with first-order logic with equality can be seen as follows. Let $\mathcal{L}$ be a first-order language with equality, $\Phi_{\mathcal{L}}$ the set of formulas of $\mathcal{L}$, and $\mathcal{M} = \langle Y, \mathfrak{r}_i \rangle_{i \in I}$ a model for $\Phi_{\mathcal{L}}$. For each formula φ of $\mathcal{L}$ let $S_{\mathcal{M}}\varphi$ be the collection of all $x \in {}^{\omega}Y$ which satisfy φ in $\mathcal{M}$. Sets $S_{\mathcal{M}}$ induce a homomorphism $\hat{S}_{\mathcal{M}} : \mathfrak{S}_{\mathcal{L}}/\equiv_\Gamma \to \mathscr{P}({}^{\omega}Y)$ with base Y. In particular, for any formula φ of $\mathcal{L}$ and any $\kappa < \omega$, we have

(9) $\quad \hat{S}_{\mathcal{M}}[v_\kappa = v_\lambda] = \mathfrak{D}_{\kappa\lambda}$

(10) $\quad \hat{S}_{\mathcal{M}}[\neg\varphi] = \overline{\hat{S}_{\mathcal{M}}[\varphi]}$

(11) $\quad \hat{S}_{\mathcal{M}}[\varphi \vee \psi] = \hat{S}_{\mathcal{M}}[\varphi] \cup \hat{S}_{\mathcal{M}}[\psi]$

(12) $\quad \hat{S}_{\mathcal{M}}[\varphi \wedge \psi] = \hat{S}_{\mathcal{M}}[\varphi] \cap \hat{S}_{\mathcal{M}}[\psi]$

(13) $\quad \hat{S}_{\mathcal{M}}[\varphi \to \psi] = \overline{\hat{S}_{\mathcal{M}}[\varphi]} \cup \hat{S}_{\mathcal{M}}[\psi]$

(14) $\quad \hat{S}_{\mathcal{M}}[(\exists v_\kappa)\varphi] = \mathfrak{C}_\kappa \hat{S}_{\mathcal{M}}[\varphi]$

---

6 Here $\mathfrak{A}$ is closed under the indicated operations and has as elements the indicated special subsets of ${}^{\alpha}Y$.

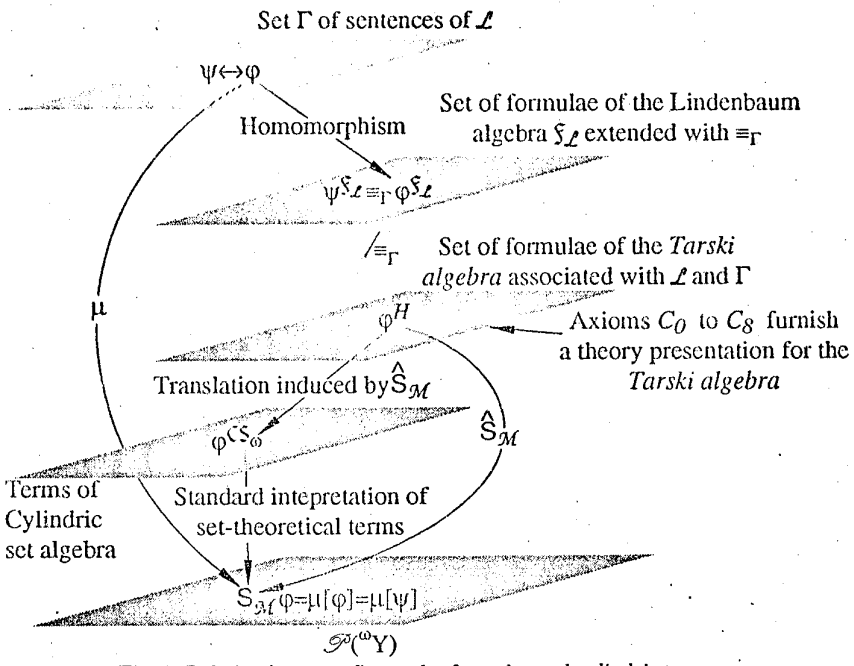

Fig. 3. Relation between first-order formulas and *cylindric* terms.

The general notion of *Cylindric Algebra* is obtained by abstraction from this set-theoretic version. So, a *Cylindric Algebra of dimension* α is a structure $\mathfrak{CA}_\alpha = \langle \mathfrak{A}, +, \bullet, -, 0, \infty, \mathsf{c}_\kappa, \mathsf{d}_{\kappa\lambda} \rangle_{\kappa,\lambda<\alpha}$ satisfying the following axioms for all $\kappa, \lambda, \mu < \alpha$ and $x, y \in \mathfrak{A}$:

C_0 $\langle \mathfrak{A}, +, \bullet, -, 0, \infty \rangle$ is a Boolean Algebra, $\mathsf{c}_\kappa : \mathfrak{A} \to \mathfrak{A}$ and $\mathsf{d}_{\kappa\lambda} \in \mathfrak{A}$

C_1 $\mathsf{c}_\kappa 0 = 0$

C_2 $x \prec \mathsf{c}_\kappa x$ [7]

C_3 $\mathsf{c}_\kappa(x \bullet \mathsf{c}_\kappa y) = \mathsf{c}_\kappa x \bullet \mathsf{c}_\kappa y$

C_4 $\mathsf{c}_\kappa \mathsf{c}_\lambda x = \mathsf{c}_\lambda \mathsf{c}_\kappa x$

C_5 $\mathsf{d}_{\kappa\kappa} = \infty$

C_6 if $\kappa \neq \lambda \neq \mu$ then $\mathsf{d}_{\lambda\mu} = \mathsf{c}_\kappa(\mathsf{d}_{\kappa\lambda} \bullet \mathsf{d}_{\kappa\mu})$

C_7 if $\kappa \neq \lambda$ then $\mathsf{c}_\kappa(\mathsf{d}_{\kappa\lambda} \bullet x) \bullet \mathsf{c}_\kappa(\mathsf{d}_{\kappa\lambda} \bullet -x) = 0$

Notice that $\mathfrak{CA}_\alpha$ is a Boolean Algebra with Operators and that $\mathfrak{Sf}_{\mathcal{L}/\equiv_\Gamma}$ is a cylindric algebra of dimension ω, which satisfies the additional condition of *locally finite dimension*:

---

7 Where $\prec$ represents the natural order within the Boolean reduct.

C_8 for any $x \in \mathfrak{A}$ there are only finitely many $\kappa \leq \lambda$ such that $c_\kappa x \neq x$

Hence, the eight axioms defining a *Cylindric Algebra of dimension* ω furnish a *theory presentation* for the *Tarski algebra* associated with $\mathcal{L}$ and Γ. A better grasp of what was discussed above can be obtained by examining figure 3. *Polyadic Algebras.* Let us, now, recall the concept of *Polyadic Algebra*, via *Quantifier* and *Transformation Algebras* [Hal62].

First, a *Quantifier Algebra of degree* χ (where χ is a cardinal number) is a triple $\langle \mathcal{B}, J, \exists \rangle$, where $\mathcal{B}$ is a *Boolean Algebra*, J is a set with cardinality χ, and $\exists$ is a function from subsets of J to quantifiers (endomorphisms) on $\mathcal{B}$, such that for all $t \in \mathcal{B}$, $\exists(\varnothing)t = t$, and $\exists(H)\exists(K)t = \exists(H \cup K)t$ whenever H, $K \subseteq J$. The set J is referred to as the index set for the variables. We are interested in algebras of degree ω actually in those whose elements correspond to formulas with finitely many variables; these are the so-called *locally finite Quantifier Algebras of degree* ω, i.e., to every $t \in \mathcal{B}$ there corresponds a finite subset H of J such that $\exists(J - H)t = t$.

Now, a *Transformation Algebra* is a triple $\langle \mathcal{B}, J, S \rangle$, where $\mathcal{B}$ is a *Boolean Algebra*, J is a set, and S is a function from transformations on J (i.e., mappings of J into itself) to endomorphisms on $\mathcal{B}$, such that for all $t \in \mathcal{B}$, $S(\delta)t = t$ where δ is the identity transformation on J, and $S(\sigma)S(\tau)p = S(\sigma\tau)p$, whenever σ and τ are transformations on J.

Finally, a *Polyadic Algebra of degree* ω is a quadruple $\mathcal{P} = \langle \mathcal{B}, J, S, \exists \rangle$, where $\langle \mathcal{B}, J, \exists \rangle$ is a *locally finite Quantifier Algebra of degree* ω and $\langle \mathcal{B}, J, S \rangle$ is a *Transformation Algebra*, such that for all $t \in A$, $S(\sigma)\exists(H)t = S(\rho)\exists(H)t$ whenever $H \subseteq J$ and σ and ρ are transformations on J agreeing on the set $J - H$, and $\exists(H)S(\rho)t = S(\rho)\exists(\rho^{-1}H)t$ whenever $H \subseteq J$ and ρ is a transformation on J never mapping any two distinct elements of J into the same element of H. We can have a better grasp of the idea behind Polyadic Algebras by looking again at figure 2. A Polyadic Algebra provides just one function $\exists$ from subsets of J to quantifiers and infinitely many transformations on J (each one allowing the construction of one of the infinitely many alfabetical arrangements of the variables).

Polyadic Algebra is then an algebra of pure first-order logic. Notice again that a Polyadic Algebra $\mathcal{P} = \langle \mathcal{B}, J, S, \exists \rangle$ is a Boolean Algebra with operators.

The Fitness of Abstract Relational Algebras, Cylindric and Polyadic Algebras to Underlying Programming Calculi. Thus, as we have seen above both Cylindric and Polyadic Algebras are Boolean Algebras with Operators, the former being an algebra for first-order logic –due to their diagonal elements– while the latter is just an algebra for pure first-order logic.

The attractive aspect of Abstract Relational Algebra for underlying a programming calculus resides in its absence of variables over individuals, the same aspect that led *functional languages* and functional based programming calculi to deserve so much attention. This absence of variables is due to the fact that the extension to Boolean Algebra necessary for constructing Abstract Relational Algebras is Peircean –i.e. finitary-. In fact, Abstract Relational Algebra is a Boolean Algebra extended with just two operators, namely, ; and $^{\mathsf{T}}$.

We should notice that both Tarski et al., with Cylindric Algebras, and Halmos, with Polyadic Algebras abandon the Peircean-like way of extending $\langle \mathcal{A}, +, \bullet, -, 0, \infty \rangle$ in favour of a more direct attack to the problem. Both approaches maintain a *Boolean Algebra* over a universe of elements whose internal structure, if any, is immaterial. By imposing an infinite arity, they produce theories over sequences of infinite length restricted by a local finiteness condition. Tarski et al. choose to represent the existential quantifier by infinitely many cylindrifications c_ζ (one for each variable v_ζ), and equations $v_\zeta = v_\xi$ by means of a doubly infinite sequence of diagonal elements $\delta_{\zeta\xi}$. Halmos, on the other hand, represents the existential quantifier by means of a single function $\exists$ but introduces infinitely many transformations on the index set J. We should notice that both *Cylindric* and *Polyadic Algebras* are models laden with a syntactical artefact, in that the former, by means of each of the "ζ-axis", and the latter, by means of the index set J, have hidden "names" for variables over individuals.

4 ∇-Extended Abstract Relational Algebra

Let us analyse here, a way for constructing a finitary algebra of first-order logic with equality introduced by A. M. Haeberer, P. A. S. Veloso, G. A. Baum and P. Elustondo [Hae90, 91, Vel91a, 91b, 92].

4.1 Tackling the Expressivennes Problem as Posed By Tarski, Henkin and Monk

First of all we will analyse and solve the expressiveness problem as it was posed by Tarski [Tar41]. So, we will try to express, by means of ∇ operation --in addition to classical relational operations-- one of the first-order expressions stated by Tarski, say (5).

Let us recall expression (5),

$$(\forall x)(\forall y)(\forall z)(\exists u)(x\,r\,u \wedge y\,r\,u \wedge z\,r\,u)$$

First, notice that there is no difficulty in expressing a simpler version of (5), like

$$(\forall x)(\forall y)(\exists u)(x\,r\,u \wedge y\,r\,u) \tag{15}$$

Since this is equivalent to

$$(\forall x)(\forall y)(\exists u)\left(x\,r\,u \wedge u\,r^{\mathsf{T}}\,y\right)$$

it can be expressed by $r ; r^{\mathsf{T}} = \infty$. The key idea here is the fact that the existential quantifier $(\exists u)$ can be simulated by the *relative* product. If we would try to apply this simple idea to (5), we would be led to something like

$$(\forall x)(\forall y)(\forall z)(\exists u)\left(x\,r\,u \wedge u\,r^{\mathsf{T}}\,y \wedge u\,r^{\mathsf{T}}\,z\right)$$

This is equivalent to (5), but we cannot simulate the effect of $(\exists u)$ by the relative product. The reason for this is the fact that variable u now occurs three times in the matrix of the formula. Tarski's relational calculus has no variables over individuals, and the relative product $r ; r^{\mathsf{T}}$ "consumes", so to speak, variable **u**.

Let us extend Proper Relational Algebra with a new operation ∇ –which we will call *fork* – defined as

$$(\forall x)(\forall y)(\forall z)\left(x\,r \nabla s\,[y,z] \leftrightarrow x\,r\,y \wedge x\,s\,z\right) \tag{16}$$

Where [ ] is a non-commutative pair formation operation. Notice that in doing so, variables over individuals will no more range over a fixed set A without any noticeable structure but over a set A^* of all finite trees made out of elements of a fixed set A, i.e., a free groupoid generated by A.

Another useful operation which can be defined from ∇ is $r \otimes s = \left((1 \nabla \infty)^{\mathsf{T}} ; r\right) \nabla \left((\infty \nabla 1)^{\mathsf{T}} ; s\right)$. It is easy to see that the Extended Proper Relational Algebra expression for such operation will be,

$$(\forall x)(\forall y)(\forall z)(\forall w)\left([x,z]\, r \otimes s\, [y,w] \leftrightarrow x\, r\, y \wedge z\, s\, w\right)$$

Then, by finitely extend the Proper Relational Algebra with the operation ∇ we obtain a Proper Algebra over locally finite trees.

Shifting back our attention to the problem of expressing relationally first-order formula (5), it is easy to see that in the same way that we were able to express (15) as $r ; r^{\mathsf{T}} = \infty$, we can now express (5) as $r ; \left(r^{\mathsf{T}} \nabla r^{\mathsf{T}}\right) = \infty \nabla \infty$.

4.2 The Expressiveness of ∇-Extended Abstract Relational Algebra

Encouraged by the result presented in Subsection 4.1 , one is tempted to conjecture that a Boolean Algebra extended with operations ; and $^{\mathsf{T}}$ as above and an abstract operation ∇ –resulting from the axiomatization of ∇ from its properties derived within the above extended Proper Relational Algebra– has the expressive power of first-order logic. We will show that indeed this is so.

Definable Subsets and their Representation. Consider a *first-order language* $\mathcal{L}$ with equality $\approx$, given by a set $P_{\mathcal{L}}$ of *predicate symbols* (we do not consider function symbols, since functions can, for our purposes, be replaced by their graphs). As usual, a *structure* $\mathfrak{B}$ for $\mathcal{L}$ consists of a *domain* B together with a *realisation* $p^{\mathfrak{B}} \subseteq {}^{\alpha}B$, for each α-ary predicate symbol p in $P_{\mathcal{L}}$. This can be extended to assign a realisation for each formula $\varphi \in \Phi_{\mathcal{L}}$. It is more convenient, however, to view a formula as defining a set of trees over B. For this purpose we introduce some notation to be also used later. We let $f(\varphi)$ be the set of variables with free occurrences in formula φ, and for a finite set Ψ of variables, we let $h(\Psi) = max\{i : v_i \in \Psi\}$. Consider a formula $\varphi \in \Phi_{\mathcal{L}}$ with $h(f(\varphi)) = n$; we associate with φ a set $\varphi^{\mathfrak{B}}$ of strings over B defined as follows

$$\varphi^{\mathfrak{B}} = \{[b_1, [\ldots, [b_{n-1}, b_n]] : \mathfrak{B} \models \varphi(b_1, \ldots, b_n)\}$$

where $\mathfrak{B} \models \varphi(b_1, \ldots, b_n)$ means that the assignment of b_i to v_i, for $i = 1, \ldots, n$, satisfies φ in $\mathfrak{B}$ (see, e.g. [Ebb80]). We call a subset $\mathcal{S}$ of $\mathcal{U} = B^*$ *definable* iff $\mathcal{S} = \varphi^{\mathfrak{B}}$ for some formula $\varphi \in \Phi_{\mathcal{L}}$. We shall denote by $\mathcal{E}(\mathfrak{B})$ the set of all finite unions of definable subsets of $\mathcal{U}$. (The reason for the closure under union is the fact that each definable subset consists of trees of the same length.) Clearly, $\mathcal{E}(\mathfrak{B})$ is a *Boolean Algebra* of subsets of $\mathcal{U}$.

A structure $\mathfrak{B}$ for $\mathcal{L}$ assigns to each n-ary predicate symbol p an n-ary relation $p^{\mathfrak{B}}$. In addition, we wish to regard a structure as assigning a binary relation (on trees) to each predicate symbol. Thus, we define the *relational realisation* of p in $\mathfrak{B}$ as $\mathfrak{B}[p] = \{\langle [b_1, \ldots, b_n], \lambda\rangle : \langle b_1, \ldots, b_n\rangle \in p^{\mathfrak{B}}\}$ (where λ is a fixed special value).

The above relational interpretation of predicate symbols suggests using them to build other relations by means of the relational operations. The set $T(\mathcal{L})$ of *relational terms* is obtained from the *basic terms*, 1, ∞, and p (for each $p \in P_{\mathcal{L}}$), by means of the

relational operation symbols +, •, ¯, ;, $^{\mathsf{T}}$, ∇. (This is the set of *closed terms*, the set of terms with variables in $\mathcal{W}$ having, in addition, all variables in $\mathcal{W}$ as basic terms.)
Now, given a structure $\mathfrak{B}$ for $\mathcal{L}$, with domain B, each such term † denotes a binary relation $\mathfrak{B}[\dagger]$ on $\mathcal{U} = B^*$ defined inductively in the obvious way. We call a binary relation $r \subseteq \mathcal{U} \times \mathcal{U}$ *denotable* iff $r = \mathfrak{B}[\dagger]$ for some † ∈ T($\mathcal{L}$), and use $\mathcal{D}(\mathfrak{B})$ to refer to the set of all denotable relations over $\mathfrak{B}$.
In order to correlate the elementary subsets in $\mathcal{E}(\mathfrak{B})$ with the denotable relations in $\mathcal{D}(\mathfrak{B})$, we need a way to represent a subset $S \subseteq \mathcal{U}$ by a binary relation $r \subseteq \mathcal{U} \times \mathcal{U}$. The relational interpretation of a predicate symbol suggests representing the set $S \subseteq \mathcal{U}$ by the relation $S \times \{\lambda\}$. Let $\Lambda(\mathcal{U})$ be the set of all binary relations r over $\mathcal{U}$ with $\mathcal{R}an(r) = \{\lambda\}$, i.e., restrictions to the converse of points λ. We define $\Lambda: \mathcal{P}(\mathcal{U}) \to \Lambda(\mathcal{U})$ by $\Lambda(S) = S \times \{\lambda\}$. This representation is quite natural, in that $\Lambda(p^{\mathfrak{B}}) = \mathfrak{B}[p]$[8]; but another one will prove more convenient. Let $I(\mathcal{U})$ be the set of all identity relations over $\mathcal{U}$ and define $I: \mathcal{P}(\mathcal{U}) \to I(\mathcal{U})$ by assigning to S the identity relation over S.
Some simple properties of these representations will be of interest. First, both representations of sets are equivalent, via a bijection $\mathsf{b}: \Lambda(\mathcal{U}) \to I(\mathcal{U})$ and its inverse. Second, these bijections can be represented by terms, b being represented by the term $\mathsf{b}(w) = (w; \infty) \bullet 1$ –see Subsection 5.1 . Thus, from the viewpoint of denotable relations, they are interchangeable. Finally, both representations are bijective, their inverses assigning to each relation its domain. In fact, it is interesting to notice that $x \in S$ iff $\langle x, x \rangle \in I(S)$. Thus, S is elementarily definable iff $I(S)$ is so, both of them by prenex formulas with the same prefix [Ebb80].

The Expressive Power of ∇-Extended Abstract Relational Algebra. We are now ready for the main result of this section, which will show that terms in ∇-Extended Algebra of Relations have the expressive power of first-order logic. This will be established by showing that any set of trees that can be defined by a first-order formula can also be denoted by a closed term of such algebra. In view of the remarks in the preceding section, concerning representation of subsets as relations, it suffices to show that, for every set in $\mathcal{E}(\mathfrak{B})$, there exists a term † in T($\mathcal{L}$), such that $\mathfrak{B}[\dagger]$ is the identity on S. So, let us denote by $\mathfrak{I}(\mathcal{L})$ the set of all terms † in T($\mathcal{L}$), such that, for every structure $\mathfrak{B}$ for $\mathcal{L}$, $\mathfrak{B}[\dagger] \subseteq \mathfrak{B}[I]$, and by $\mathfrak{I}(\mathfrak{B})$ the set of their denotations in structure $\mathfrak{B}$ for $\mathcal{L}$. Also we define, by induction on m, $\Theta_1 = I_B$ and $\Theta_m = \Theta_{m-1} \times I_B$, where I_B is the restriction of I to B.

Proposition 6 Given a first-order language $\mathcal{L}$, there exists a function $\mathcal{T}: \Phi_{\mathcal{L}} \to \mathfrak{I}(\mathcal{L})$, such that for every structure $\mathfrak{B}$ for $\mathcal{L}$, $I(\varphi^{\mathfrak{B}}) = \mathfrak{B}[\mathcal{T}(\varphi)]$, for every formula $\varphi \in \Phi_{\mathcal{L}}$.

Proof We will define $\mathcal{T}$ by induction on the structure of formula φ.

Basis: We must distinguish three cases.

---

8 In particular, making λ the truth value T (true) this representation was adopted by M. Frias and R. Wachenchauzer [Fri92], who used the Haeberer-Veloso programming calculus [Hae91] for deriving and optimising programs which interact with relational data bases.

Case 1: φ is $v_1 \approx v_2$. Then, we set

$$\mathcal{T}(v_1 \approx v_2) = (I \nabla I)^{\mathsf{T}} ; I_B ; (I \nabla I)$$

Case 2: φ is $p(v_1, \ldots, v_n)$ with $p \in P_{\mathcal{L}}$. Then, we set

$$\mathcal{T}(p(v_1, \ldots, v_n)) = \mathbf{b}(p)^{\mathsf{T}} ; \mathbf{b}(p) = (p ; p^{\mathsf{T}})) \bullet I$$

where $\mathbf{b}(p) = (p ; \infty) \bullet I$ denotes the conversion of $\mathfrak{B}[p]$ to $I(p^{\mathfrak{B}})$.

Case 3: φ is an atomic formula $\alpha(u_1, \ldots, u_m)$ obtained from one of the above $\alpha(v_1, \ldots, v_n)$ by means of a substitution σ. Then, Lemma 8 below yields

$$\mathcal{T}(\alpha(u_1, \ldots, u_m)) = s^{\mathsf{T}} ; \mathcal{T}(\alpha(v_1, \ldots, v_n)); s$$

where s^{T} is a term, simulating σ, easily constructed by means of I, I_B and ∇.

Inductive step: We must distinguish two cases

If φ is $\neg\psi$, then we set $\mathcal{T}(\neg\psi) = \overline{\mathcal{T}(\neg\psi)} \bullet e_n$

Now, let $h(f(\varphi)) = n$. In view of Lemma 7 below, it suffices to consider φ' of the form $\varphi \wedge v_1 \approx v_1 \wedge \ldots \wedge v_n \approx v_n$.

If φ is $\psi \vee \theta$, then φ' is equivalent to $\psi' \vee \theta'$, where ψ' is $\psi \wedge v_1 \approx v_1 \wedge \ldots \wedge v_n \approx v_n$ and similarly for θ'. Lemma 1 applied to the inductive hypothesis gives $\mathcal{T}(\psi')$ and $\mathcal{T}(\theta')$. We then set

$$\mathcal{T}(\psi \vee \theta) = \mathcal{T}(\psi') + \mathcal{T}(\theta')$$

If φ is $(\exists v_i)\psi$, clearly φ' is equivalent to $(\exists v_n)\psi' \wedge v_n \approx v_n$, where ψ' is obtained from ψ by the substitution σ that interchanges v_i and v_n. So, Lemma 8 below, applied to the inductive hypothesis, gives $\mathcal{T}(\psi')$. Then, we set:

$$\mathcal{T}((\exists v_n)\psi') = (I \nabla \infty) ; \mathcal{T}(\psi') ; (I \nabla \infty)^{\mathsf{T}}$$

whence Lemma 7 will give $\mathcal{T}(\varphi)$.

Q. E. D.

Lemma 7 For every $n \geq 0$, there exists a term d_n, such that, for every formula φ with $h(f(\varphi)) = n$, if φ' is the formula $\varphi \wedge v_1 \approx v_1 \wedge \ldots \wedge v_{m+n} \approx v_{m+n}$, one can have $\mathcal{T}(\varphi') = \mathcal{T}(\varphi) \otimes e_m$ and $\mathcal{T}(\varphi) = d_n^{\mathsf{T}} ; \mathcal{T}(\varphi') ; d_n$.

Proof Trivial by induction using $(I \nabla \infty)^{\mathsf{T}}$, $(\infty \nabla I)^{\mathsf{T}}$ and ∇, see [Vel91b].

Lemma 8 Given a substitution σ on $\{1, \ldots, n\}$, there exists a term $s(\sigma)$, such that for any formula φ with $h(f(\varphi)) = n$, if $\sigma(\varphi)$ is the formula obtained by applying σ to φ, then one can take $\mathcal{T}(\sigma(\varphi)) = s(\sigma)^{\mathsf{T}} ; \mathcal{T}(\varphi) ; s(\sigma)$.

Proof Trivial, see [Vel91b] and the discussion on the construction of non-atomic cylindrifications below.

Now, it is easy to see that $\mathfrak{I}(\mathcal{L}) = \{t \bullet I : t \in \mathcal{T}(\mathcal{L})\}$. So, from the very definitions of the operations and constants of the ∇-Extended Relational Algebra one can see that, for each term $t \in \mathfrak{I}(\mathcal{L})$, one has formulae $\varphi_1, \ldots, \varphi_n \in \Phi_{\mathcal{L}}$ defining its domain (in the sense that for every structure $\mathfrak{B}$ for $\mathcal{L}$, $Dom(\mathfrak{B}[t]) = \varphi_1^{\mathfrak{B}} \cup \ldots \cup \varphi_n^{\mathfrak{B}}$). Thus, we have our main result:

Theorem 9 Given a first-order language $\mathcal{L}$ with equality and a structure $\mathfrak{B}$ for $\mathcal{L}$, $I(\mathcal{E}(\mathfrak{B})) = \mathfrak{I}(\mathfrak{B})$. Hence, for every $\mathcal{S} \subseteq B^*$, there exists $\dagger \in T(\mathcal{L})$, such that $\mathfrak{B}[\dagger] = I(\mathcal{S})$ iff $\mathcal{S} \in \mathcal{E}(\mathfrak{B})$;and conversely, for every $\dagger \in \mathcal{T}(\mathcal{L})$, there exists $\mathcal{S} \in \mathcal{E}(\mathfrak{B})$ such that $\mathfrak{B}[\dagger] = I(\mathcal{S})$ iff $\dagger \in \mathfrak{I}(\mathfrak{B})$.

Therefore, the representation of every definable subset of $\mathfrak{B}$ is denoted by a *relational term.*.

Notice that in Proposition 6 we define the translation $\mathcal{T}$ for the existential quantifier just for the case of the quantification on variable v_n. For generalising the quantification on other variables we relay on terms d_n and $s(\sigma)$ for respectively adjusting the number of variables and producing an alphabetical transformation of the original formula. This seems to be very close to the way Polyadic Algebras treat the same problem, the difference being that, in case of ∇-Extended Relational Algebra, this trick is an artefact of the proof, while in the former it is inherent in the algebra itself. Notice also that both d_n and $s(\sigma)$ are constructed with the constants and operations of the ∇-Extended Relational Algebra.

By calling $\pi = (1 \nabla \infty)^{\mathsf{T}}$ and $\rho = (\infty \nabla 1)^{\mathsf{T}}$ and using a somewhat two-dimensional notation we can express, for instance, $\mathcal{T}((\exists v_1)(\exists v_3)\psi(v_0, v_1, v_2, v_3, v_4))$ as,

$$\underbrace{\begin{pmatrix} & \begin{pmatrix}\pi;\pi\\ \nabla\\ \rho\end{pmatrix}^{\mathsf{T}} \\ \pi;\pi; & \nabla \\ & \rho \end{pmatrix}}_{ds} ; \mathcal{T}\left(\psi(v_0, v_1, v_2, v_3, v_4)\right); \underbrace{\begin{pmatrix} & \begin{pmatrix}\pi;\pi\\ \nabla\\ \rho\end{pmatrix} \\ \pi;\pi; & \nabla \\ & \rho \end{pmatrix}}_{ds^{\mathsf{T}}}$$

where the pair of terms $\langle ds, ds^{\mathsf{T}} \rangle$ is the construction, within ∇-Extended Relational Algebra, of the cylindrification $c_1 c_3\left(\psi^H(v_0, v_1, v_2, v_3, v_4)\right)$.

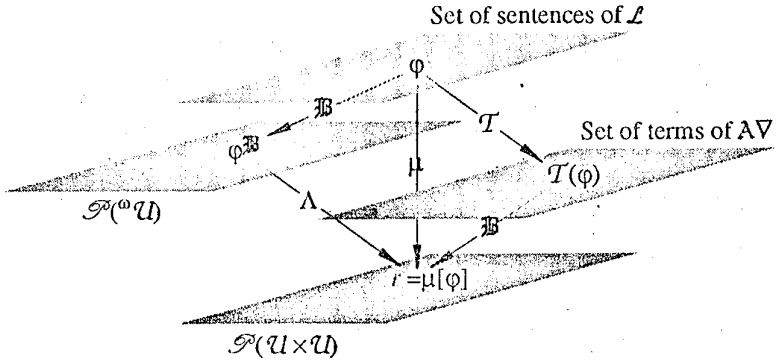

Fig. 4. Domains and morphisms involved in the proof of expressiveness of ∇-Extended Relational Algebra

4.3 An Axiomatization of ∇-Extended Abstract Relational Algebra.
Let us now introduce an axiomatization for ∇-Abstract Relational Algebra.

Definition 10 A *∇-Extended Abstract Relational Algebra* is a structure $A\nabla = \langle \Re, +, \bullet, ^-, ;, ^\mathsf{T}, \nabla \rangle$ over a non-empty set $\Re$, such that

(17) $\langle \Re, +, \bullet, ^- \rangle$ is a complete atomic Boolean Algebra. Its *zero element* will be denoted by *0*, and its *unit element* by ∞. The symbol $\subset$ is used for the ordering with respect to the lattice structure and is called *inclusion*

(18) $\langle \Re, ; \rangle$ is a semigroup with exactly one *identity element* which is denoted by *1*, i.e. $(r;s);t = r;(s;t)$ & $r;1 = 1;r = r$

(19) $r;s \subset t \leftrightarrow r^\mathsf{T};\bar{t} \subset \bar{s} \leftrightarrow \bar{t};s^\mathsf{T} \subset \bar{r}$ *Schröder rule*

(20) $r \neq 0 \rightarrow \infty;r;\infty = \infty$ *Tarski rule*

(21) $r \nabla s = (r;(1 \nabla \infty)) \bullet (s;(\infty \nabla 1))$

(22) $(r \nabla s);(t \nabla q)^\mathsf{T} = (r;t^\mathsf{T}) \bullet (s;q^\mathsf{T})$

(22') $\left((1 \nabla \infty)^\mathsf{T} \nabla (\infty \nabla 1)^\mathsf{T}\right) + 1 = 1$

Notice that in the preceding section we call, for the sake of compactness, $\pi = (1 \nabla \infty)^\mathsf{T}$ and $\rho = (\infty \nabla 1)^\mathsf{T}$. The intention behind the choice of symbols π and ρ is to connect $(1 \nabla \infty)^\mathsf{T}$ and $(\infty \nabla 1)^\mathsf{T}$ with the first and second projections respectively.
The concept of *projections* and of some kind of *product* in connection with algebras of first-order logic and, in special, with Relational Algebras appears as early as 1946 with the seminal work of Everett and Ulam on *Projective Algebra* [Eve46]. Later on, De Roever [Roe72], Schmidt and Ströhlein [Sch85b], Zierer [Zie83], Berghammer and Zierer [Ber86], Berghammer [Ber91], and Backhouse et al. [Bac92][9], introduced the product and projections either as data types or as operations. However, each one of them failed to formulate that a Relational Algebra extended with a product of this kind has enough expressive power. Everett and Ulam (and, later, Bednarek and Ulam [Bed78]) when defining Projective Algebra, extended Boolean Algebra with three fundamental operations: two projections p_1 and p_2 and a product □. In doing so they failed by not recognising the power of their combination with Peircean operations for the construction of a Relational Algebra over locally finite trees. Those researchers involved in computer science failed because they had a categorical viewpoint that "masked" the non fundamental character of projections.
Formally speaking we can extend algebra A∇ by definition as,

Definition 11

(23) $\pi = (1 \nabla \infty)^\mathsf{T}$

(24) $\rho = (\infty \nabla 1)^\mathsf{T}$

---

9 The idea of direct product and direct sum presented in this paper by Backhouse et al. was borrowed from the works of the Munich group.

(25) $r \otimes s = \left((1 \nabla \infty)^{\mathsf{T}} ; r\right) \nabla \left((\infty \nabla 1)^{\mathsf{T}} ; s\right)$

Comparing our Axiomatization with Other Approaches to Projections and Product. We are able now to compare our axiomatization with both, the Munich one and that due to Backhouse et al.

First let us prove that the Munich group[10] axiomatization can be derived from our one. The Munich axiomatization is, in our own notation,

M_0 $\quad \pi^{\mathsf{T}} ; \pi = 1$

M_1 $\quad \rho^{\mathsf{T}} ; \rho = 1$

M_2 $\quad (\pi ; \pi^{\mathsf{T}}) \bullet (\rho ; \rho^{\mathsf{T}}) = 1$

M_3 $\quad \pi^{\mathsf{T}} ; \rho = \infty$ and $\rho^{\mathsf{T}} ; \pi = \infty$[11]

M_4 $\quad r \nabla s = (r ; \pi^{\mathsf{T}}) \bullet (s ; \rho^{\mathsf{T}})$

M_0 can be derived as follows, by applying (23), whereby we can write $(1 \nabla \infty) ; (1 \nabla \infty)^{\mathsf{T}}$, which by axiom (22) equals $(1 ; 1^{\mathsf{T}}) \bullet (\infty ; \infty^{\mathsf{T}}) = 1$. M_1 can be proved similarly. To derive M_2 notice that from (25) one has $1 \otimes 1 = \left((1 \nabla \infty)^{\mathsf{T}} ; 1\right) \nabla \left((\infty \nabla 1)^{\mathsf{T}} ; 1\right) = (1 \nabla \infty)^{\mathsf{T}} \nabla (\infty \nabla 1)^{\mathsf{T}}$ which by applying axiom (21) can be rewritten as $\left((1 \nabla \infty)^{\mathsf{T}} ; (1 \nabla \infty)\right) \bullet \left((\infty \nabla 1)^{\mathsf{T}} ; (\infty \nabla 1)\right)$ which by (23) and (24) is $(\pi ; \pi^{\mathsf{T}}) \bullet (\rho ; \rho^{\mathsf{T}})$ – notice that we proved equality with $1 \otimes 1$ because we are working with homogeneous relations. M_3 can be derived in a straightforward way from (23), (24) and (22). Finally, M_4 is equal to (21), provided that we use definitions (23) and (24).

As for the axiomatization due to Backhouse et al. we can show that it exhibits more axioms than are necessary. As presented in [Bac92] it looks like,

B_0 $\quad r \nabla s = (r ; \pi^{\mathsf{T}}) \bullet (s ; \rho^{\mathsf{T}})$

B_1 $\quad (\pi ; \pi^{\mathsf{T}}) \bullet (\rho ; \rho^{\mathsf{T}}) \subset 1$

B_2 $\quad (r \nabla s) ; (t \nabla q)^{\mathsf{T}} = (r ; t^{\mathsf{T}}) \bullet (s ; q^{\mathsf{T}})$

B_3 $\quad \mathit{Dom}(\pi) = \mathit{Dom}(\rho)$

Notice that B_0 equals (21), B_1 equals M_2 in the homogeneous case , B_2 equals (22) and B_1 equals (22').

As for B_3 let us introduce the following definition.

Definition 12 $\quad 1_{\mathit{Dom}(r)} = (r ; r^{\mathsf{T}}) \bullet 1$

Then, B_3 can be derived immediately from M_2 and Definition 12 provided the equivalence $\mathit{Dom}(r) = \mathit{Dom}(s) \leftrightarrow 1_{\mathit{Dom}(r)} = 1_{\mathit{Dom}(s)}$ holds, which in turn, is obvious, since 1_X is the equivalence relation restricted to set X. Here, we are using the usual meaning for $\mathit{Dom}(r)$, i.e., $(\forall x)\left((x \in \mathit{Dom}(r)) \leftrightarrow (\exists y)(xry)\right)$.

---

10 Schmidt, Ströhlein, Zierer and Berghammer.

11 Actually one of them suffices for this axiomatization, we include both just for symmetry.

4.4 On the Representability of ∇-Extended Abstract Relational Algebra

The representability of ∇-Extended Abstract Relational Algebra as defined by axioms (17) to (22) is still an open problem, being the object of ongoing research. However, as was discussed in [Bau92], ∇-Extended Abstract Relational Algebra is representable if we add to the above mentioned axioms the following one,

(26) $t \neq 0 \wedge t \subset \infty \nabla \infty \rightarrow (\exists v)(\exists w)(0 \neq v \nabla w \subset t)$

Then, denoting by $\mathscr{A}$ the set of atoms of A ∇, representability can be proved in the following way:

Proposition 13 $\alpha \in \mathscr{A} \rightarrow \alpha \nabla \alpha \in \mathscr{A}$.

Proof Take β such that $\beta \neq 0$ and $\beta \subset \alpha \nabla \alpha$, then, by axiom (26), there exist γ and δ such that $0 \neq \gamma \nabla \delta \subset \beta$ and consequently $\gamma \neq 0 \neq \delta$.
Now multiplying by projections, we obtain
$\gamma \bullet (\delta ; \infty) = (\gamma \nabla \delta) ; \pi \subset \beta ; \pi \subset (\alpha \nabla \alpha) ; \pi = \alpha \bullet (\alpha ; \infty) = \alpha$
$(\gamma ; \infty) \bullet \delta = (\gamma \nabla \delta) ; \rho \subset \beta ; \rho \subset (\alpha \nabla \alpha) ; \rho = (\alpha ; \infty) \bullet \alpha = \alpha$
Because $\alpha \in \mathscr{A}$ and $\gamma \bullet (\delta ; \infty) \neq 0$, we have $\gamma \bullet (\delta ; \infty) = \alpha$.
Therefore, $\alpha \subset \gamma$ and $\alpha \subset \delta$, giving $\alpha \nabla \alpha \subset \gamma \nabla \delta \subset \beta$ so that indeed $\beta = \alpha \nabla \alpha$

Q. E. D.

Proposition 14 $\alpha \in \mathscr{A} \rightarrow (\alpha ; \overline{1}) \bullet \alpha = 0$.

Proof From (22) we can deduce that $(\alpha ; \overline{1}) \bullet \alpha = (\alpha \nabla \alpha) ; (\overline{1} \nabla 1)^{\mathsf{T}}$
since $r ; (s \nabla t) \subset r ; s \nabla r ; t$ [Hae91] we have $\alpha ; (1 \nabla 1) \subset \alpha \nabla \alpha$. But, by hypothesis and Proposition 13, $\alpha \nabla \alpha \in \mathscr{A}$, so that we have $\alpha ; (1 \nabla 1) = \alpha \nabla \alpha$ (since $\alpha ; (1 \nabla 1)$ cannot be 0).
Then, $(\alpha ; \overline{1}) \bullet \alpha = (\alpha \nabla \alpha) ; (\overline{1} \nabla 1)^{\mathsf{T}} = \alpha ; \left((\overline{1} \nabla 1) ; \left((1 \nabla 1)^{\mathsf{T}} \right) \right)$, which, by (22), equals $\alpha ; (\overline{1} \bullet 1)$, thus, $(\alpha ; \overline{1}) \bullet \alpha = 0$.

Q. E. D.

Lemma 15 $(r ; s) \bullet t = \left(r ; \left((r^{\mathsf{T}} ; t) \bullet s \right) \right) \bullet t$.

Proof See [Sch91] page 21.

Proposition 16 $\alpha \in \mathscr{A} \rightarrow (\alpha^{\mathsf{T}} ; \alpha) \bullet \overline{1} = 0$.

Proof By applying Lemma 15 we have $(\alpha^{\mathsf{T}} ; \alpha) \bullet \overline{1} = \left(\alpha^{\mathsf{T}} ; \left((\alpha ; \overline{1}) \bullet \alpha \right) \right) \bullet \overline{1}$
But, from Proposition 14, we have $\alpha \in \mathscr{A} \rightarrow (\alpha ; \overline{1}) \bullet \alpha = 0$, thus, $(\alpha^{\mathsf{T}} ; \alpha) \bullet \overline{1} = 0$

Q. E. D.

Proposition 17 $\alpha \in \mathscr{A} \rightarrow \alpha$ *is functional.*

Proof By Proposition 16 $\alpha \in \mathscr{A} \rightarrow (\alpha^{\mathsf{T}} ; \alpha) \bullet \overline{1} = 0$ and by hypothesis $(\alpha^{\mathsf{T}} ; \alpha) \subset 1$, which is a characterisation of the functionality of α [Jón52]

Q. E. D.

Hence, by a well-known result [Jón52], since its atoms are functional, the ∇-Extended Abstract Relational Algebra extended with axiom (26) is representable, i.e., all its models are isomorphic to the Proper ∇-Extended Relational Algebra.

5 On the Calculation of Recursive Expressions for Quantified Relational Terms

5.1 Some Useful Results on Partial Identities

As it should be noticed, in constructing the ∇-Extended Abstract Relational Algebra we chose to work in a homogeneous framework (for a comparative discussion about both homogeneous and heterogeneous approaches see [Ber93]). Thus, we are forced to have some kind of type relativization mechanism. It is obvious that such mechanism depends on the way we have chosen for internalising sets in the relational framework. Some authors, as Hoare and He [Hoa86a] use vectors (which they call *conditions*) and, therefore, intersection as relativizing operation. Others, as ourselves, chose partial identities, therefore using left multiplication (;) for relativizing[12] relations.

Proposition 18 shows the equivalence between both ways of relativization.

Proposition 18 $(1 \bullet (s ; \infty)) ; r = r \bullet (s ; \infty)$.

Proof It is known [Sch91] that $(q \bullet (r ; \infty)) ; s = (q ; s) \bullet (r ; \infty)$

Then, $(1 \bullet (s ; \infty)) ; r = (1 ; r) \bullet (s ; \infty) = r \bullet (s ; \infty)$

Q. E. D.

The next two propositions show that in restricting partial identities with partial identities, left multiplication and intersection are interchangeable.

Proposition 19 $1_X \bullet 1_Y = 1_X ; 1_Y$

Proof $1_X ; 1_Y = (1 \bullet (1_X ; \infty)) ; 1_Y$ (see Proposition 22 below)

By Proposition 18 we have,

$$(1 \bullet (1_X ; \infty)) ; 1_Y = 1_Y \bullet (1_X ; \infty) = 1_Y \bullet (1_X ; (1 + \overline{1})) =$$

$$1_Y \bullet (1_X + 1_X ; \overline{1}) = 1_X \bullet 1_Y$$

Q. E. D.

Proposition 20 $(1_X ; r) \bullet (1_Y ; r) = (1_X \bullet 1_Y) ; r$

---

12 For a discussion about different ways of internalising sets in a relational framework see [Vel92].

Proof $(1_X ; r) \circ (1_Y ; r) = (1_X + 1_{\overline{X}}) ; (1_X ; r) \circ (1_Y ; r) =$

$(1_X ; 1_X ; r) \bullet (1_X ; 1_Y ; r) + (1_{\overline{X}} ; 1_X ; r) \bullet (1_{\overline{X}} ; 1_Y ; r) =$

$(1_X ; r) \bullet ((1_X ; 1_Y) ; r) + ((1_{\overline{X}} ; 1_X) ; r) \bullet ((1_{\overline{X}} ; 1_Y) ; r) =$

$(1_X ; r) \bullet ((1_X ; 1_Y) ; r) = (1_X \bullet 1_Y) ; r$

Q. E. D.

In the sequel, we introduce some useful and necessary results for dealing with partial identities. Proposition 21 gives different ways of expressing $1_{Dom(r)}$ (recall Definition 12), while Proposition 22 shows in the abstract environment, the obvious proper algebraic result that any part of the identity relation is itself a partial identity.

Proposition 21 $(r ; r^{\mathsf{T}}) \bullet 1 = (r ; \infty) \bullet 1 = (1 \nabla r) ; \pi$.

Proof $(r ; \infty) \bullet 1 = (r ; (r^{\mathsf{T}} + \overline{r^{\mathsf{T}}})) \bullet 1 = (r ; r^{\mathsf{T}}) \bullet 1 + (r ; \overline{r^{\mathsf{T}}}) \bullet 1$

Recalling that $(\overline{r^{\mathsf{T}}} ; 1) \bullet r^{\mathsf{T}} = 0 \rightarrow (r ; \overline{r^{\mathsf{T}}}) \bullet 1 = 0$ (see [Tar41])
then $(r ; r^{\mathsf{T}}) \bullet 1 = (r ; \infty) \bullet 1$
Moreover,

$(1 \nabla r) ; \pi = (1 \nabla r) ; (1 \nabla \infty)^{\mathsf{T}} = (r ; \infty) \bullet 1$

Q. E. D.

Proposition 22 $d \subset 1 \rightarrow 1 \bullet (d ; \infty) = d$.

Proof $1 \bullet (d ; \infty) = 1 \bullet (d ; (1 + \overline{1})) = 1 \bullet d + 1 \bullet (d ; \overline{1}) = d$

Q. E. D.

The next two propositions show that for a given relation r, the relation $1_{Dom(r)}$ is exactly its left identity. Proposition 23 states that given any relation, the partial identity over its domain contains its left identity; while Proposition 24 shows that any part of an identity relation which behaves as left identity of a given relation, contains the identity over its domain.

Proposition 23 $1_{Dom(r)} ; r = r$

Proof See [Sch91] page 20.

Proposition 24 $1_X ; r = r \rightarrow (r ; \infty) \bullet 1 \subset 1_X$

Proof By ; monotonicity $1_X ; r = r \rightarrow 1_X ; r ; \infty = r ; \infty$

by $\bullet$ monotonicity $(1_X ; r ; \infty) \bullet 1 = (r ; \infty) \bullet 1$

but $(1_X ; r ; \infty) \bullet 1 \subset (1_X ; \infty) \bullet 1 = 1_X ; 1 = 1_X$ then $(r ; \infty) \bullet 1 \subset 1_X$

Q. E. D.

5.2 On Refinements

It is noticeable in calculating programs that equality is a too narrow substitution criterion while inclusion is obviously too wide. What we need is to introduce a refinement order relation among relations. So, let us state the following,

Definition 25 $r \sqsubset s$ iff $1_{\mathcal{D}om(s)} ; r \subset s$ and $1_{\mathcal{D}om(s)} \subset 1_{\mathcal{D}om(r)}$

Its "programming" meaning is obvious, given two specifications (or programs) σ_1 and σ_2, we will say that σ_2 is totally correct with respect to σ_1 iff

- within its precondition, σ_1 satisfies σ_2 –not necessarily with the same degree of non-determinism, and
- the set denoted by the precondition of σ_2 is included in that denoted by the precondition of σ_1. Formally, $\mu[\sigma_2] \sqsubset \mu[\sigma_1]$ (see figure (4)).

Proposition 26 is also obvious in Proper Relational Algebra, but we should prove it in the abstract framework to be used itself in other proofs.

Proposition 26 $\mathcal{D}om(r) ; \infty = r ; \infty$

Proof (a) $(r ; \infty) \bullet 1 \subset r ; \infty \rightarrow \big((r ; \infty) \bullet 1\big) ; \infty \subset r ; \infty ; \infty = r ; \infty$

(b) $r ; \infty = \big((r ; \infty) \bullet 1\big) ; r ; \infty \subset \big((r ; \infty) \bullet 1\big) ; \infty$

From (a) and (b) by monotonicity of ; and since $r ; \infty = \infty$, we have $\mathcal{D}om(r) ; \infty = r ; \infty$

Q. E. D.

Proposition 27 shows how to relativize via fork and projections.

Proposition 27 $(r \nabla s) ; \pi = 1_{\mathcal{D}om(s)} ; r.$

Proof $(r \nabla s) ; \pi = (s ; \infty) \bullet r = \big(\big((s ; \infty) \bullet 1\big) ; s ; \infty\big) \bullet r$ by Proposition 23

Let $t = \overline{(s ; \infty) \bullet 1} \bullet 1$. Since for every X, Y is $1_X \bullet 1_Y = 0 \leftrightarrow 1_X ; 1_Y = 0$, it is clear that $\big((s ; \infty) \bullet 1\big) ; t = 0$

Moreover,

$\Big(s + \overline{(s ; \infty) \bullet 1} \bullet 1\Big) ; \infty = \Big(s + \big(\overline{s ; \infty} + \overline{1}\big) \bullet 1\Big) ; \infty = \Big(s + \overline{s ; \infty} \bullet 1\Big) ; \infty =$

$\Big(\big((s ; \infty) \bullet 1\big) ; s + \overline{s ; \infty} \bullet 1\Big) ; \infty = \Big(\big(s ; \infty + \overline{s ; \infty}\big) \bullet 1\Big) ; (s + 1) ; \infty =$

$(\infty \bullet 1) ; (s + 1) ; \infty = (s + 1) ; \infty = \infty$

Therefore,

$(s ; \infty) \bullet r = \big(\big((s ; \infty) \bullet 1\big) ; (s + t) ; \infty\big) \bullet r = \big(\big((s ; \infty) \bullet 1\big) ; \infty\big) \bullet r =$

$\big((s ; \infty) \bullet 1\big) ; r = 1_{\mathcal{D}om(s)} ; r$

Q. E. D.

Finally, Proposition 28 and its corollaries are an example of a refinement valid under certain conditions (this result will be used in the derivation presented in Subsection 5.5).

Proposition 28 a) $\mathcal{D}om(r\,;s)=\mathcal{D}om(r\,;\mathcal{D}om(s))$

b) $\mathcal{D}om(r\,\nabla\,s)=\mathcal{D}om(r)\bullet\mathcal{D}om(s)$

Proof (a) By Proposition 26 $(r\,;s\,;\infty)\bullet 1=\big(r\,;\big((s\,;\infty)\bullet 1\big);\infty\big)\bullet 1$
(b) By Definition 12 $\mathcal{D}om(r\,\nabla\,s)=\big((r\,\nabla\,s)\,;(r\,\nabla\,s)^{\mathsf{T}}\big)\bullet 1$,
which by applying axiom (22) we can write,
$\big(r\,;r^{\mathsf{T}}\big)\bullet\big(s\,;s^{\mathsf{T}}\big)\bullet 1=\mathcal{D}om(r)\bullet\mathcal{D}om(s)$
Q. E. D.

Corollary 29 If $\mathcal{D}om(s)=\mathcal{D}om(t)$ then

a) $\mathcal{D}om(s\,\nabla\,t)=\mathcal{D}om(s)$

b) $\mathcal{D}om(r\,;s)=\mathcal{D}om(r\,;t)$

c) $\mathcal{D}om(r\,;s)=\mathcal{D}om(r\,;s\,\nabla\,r\,;t)=\mathcal{D}om\big(r\,;(s\,\nabla\,t)\big)$

Proof Trivial using Proposition 28

Corollary 30 If $\mathcal{D}om(r\,;s)=\mathcal{D}om(r\,;t)$ then $r\,;(s\,\nabla\,t)\subseteq(r\,;s)\,\nabla\,(r\,;t)$

Proof Trivial

5.3 Calculating Algorithms From First-Order Specifications Involving Existential Quantifiers

One of the ways of translating first-order specifications onto terms of the ∇-Extended Abstract Relation Algebra is by the simple and straightforward application of the expressiveness Theorem 9. However, by doing so we obtain expressions that, from the view point of algorithmic complexity, are extremely inefficient. The ability of expressing quantifiers not by fundamental operations but by terms written from such operations, in this cases, helps dealing with such complexity.

To illustrate this claim, let us calculate a very simple and "classic" program from a non constructive first-order specification, i.e., a program for deciding whether a natural number n is of the form $2^i - 1$ for some $i \in$ Nat.

Let us first introduce some relational constants (recall that $\mathcal{U}$ is the universe of discourse of the Proper Algebra),

- *zero* $=\{\langle x,y\rangle : x\in\mathcal{U}\wedge y\in \text{Nat}\wedge y=0\}$ i.e., the converse of the point zero
- *suc* $=\{\langle x,y\rangle : x,y\in \text{Nat}\wedge y=x+1\}$
- *pred* $=\{\langle x,y\rangle : x,y\in \text{Nat}\wedge x\neq 0\wedge y=x-1\}$
- *pot*2 $=\{\langle x,y\rangle : x,y\in \text{Nat}\wedge y=2^x\}$
- *true* $=\{\langle x,y\rangle : x\in\mathcal{U}\wedge y\in \text{Bool}\wedge y=\mathsf{T}\}$ i.e., the converse of the Boolean point T
- *false* $=\{\langle x,y\rangle : x\in\mathcal{U}\wedge y\in \text{Bool}\wedge y=\mathsf{F}\}$ i.e., the converse of the Boolean point F

- $and = \{\langle [x,y],z\rangle : x,y,z \in \mathrm{Bool} \wedge z = x \wedge y\}$

where Nat is the type *natural numbers* and Bool is the type *Boolean*.
Let us now give a first-order specification of our problem,

$$\varphi(n) = (\exists i)\left(\underbrace{i \in \mathrm{Nat} \wedge n = 2^i - 1}_{\alpha(n,i)}\right)$$

Recall that, by Theorem 9 is $\mathcal{T}\big((\exists y)(\psi(x,y))\big) = \pi^{\mathsf{T}}; \mathcal{T}\big(\psi(x,y)\big); \pi$. Since, as it is obvious, $\mathcal{T}(\alpha(n,i)) = \big(1 \otimes (pot2; pred)\big); (1 \nabla 1)^{\mathsf{T}}; (1 \nabla 1)$, we can write 1_{prop} (i.e., the identity relation which interprets $\varphi(n)$) as

$$1_{prop} = 1_{Nat}; \pi^{\mathsf{T}}; \big(1 \otimes (pot2; pred)\big); (1 \nabla 1)^{\mathsf{T}}; (1 \nabla 1); \pi$$

But $(1 \nabla 1)^{\mathsf{T}}; (1 \nabla 1); \pi = (1 \nabla 1)^{\mathsf{T}}; (1 \nabla 1); (1 \nabla \infty)^{\mathsf{T}}$ and by applying axiom (22), we can write $(1 \nabla 1)^{\mathsf{T}}; 1 \bullet \infty = (1 \nabla 1)^{\mathsf{T}}; 1 = (1 \nabla 1)^{\mathsf{T}}$. Thus, we have

$$1_{prop} = 1_{Nat}; \pi^{\mathsf{T}}; \big(1 \otimes (pot2; pred)\big); (1 \nabla 1)^{\mathsf{T}}$$

Now, for a reason that will become apparent soon, we will write

$$\dot{1}_{prop} = 1_{Nat}; \pi^{\mathsf{T}}; \big(1 \otimes (pot2; pred)\big); (1 \nabla 1)^{\mathsf{T}}; true$$

Let us leave now the calculation for a moment for introducing a new operation on relations (i.e., let us do an extension by definition on the ∇-Abstract Relational Algebra). The operation we have in mind will be denoted by $r[\![s]\!]$ –which we read as *the substitution of s into r*. The result of applying $r[\![s]\!]$ is the relation obtained by replacing the arcs of r beginning in points belonging to the intersection of the domains of r and s by arcs of s (see Figure 5)

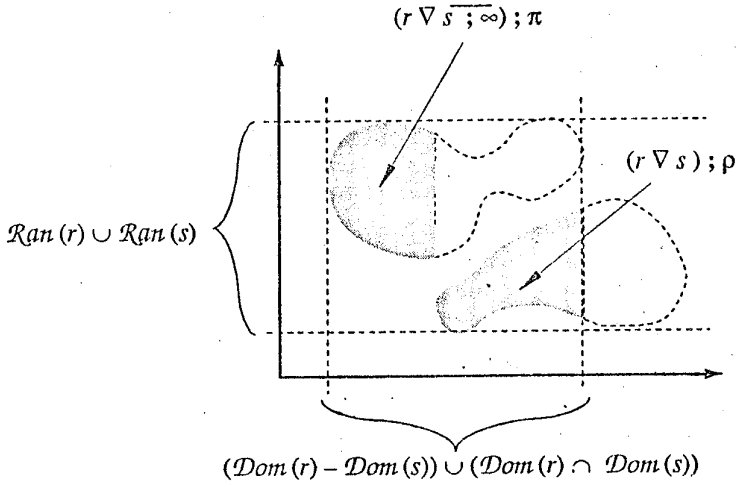

Fig. 5. $r[\![s]\!]$

Thus, substitution can be expressed as, $r[\![s]\!] = (r \nabla s); \rho + \big(r \nabla \overline{s\,;\infty}\big); \pi$.
Returning to our formal development, we can calculate a *total predicate* from the partial one $\dot{1}_{prop}$, as

$$\ddot{I}_{prop} = (I_{Nat} ; false)[\![\dot{I}_{prop}]\!] = \left((I_{Nat} ; false) \nabla \dot{I}_{prop}\right) ; \rho + \left((I_{Nat} ; false) \nabla \overline{\dot{I}_{prop} ; \infty}\right) ; \pi$$

which can be written

$$\ddot{I}_{prop} = I_{Nat} ; \pi^{\mathsf{T}} ; \left(I \otimes (pot2 ; pred)\right) ; (I \nabla I)^{\mathsf{T}} ; true + I_{\overline{prop}} ; false$$

Now, by taking into account that $(\exists i)(n = 2^i - 1) \leftrightarrow (\exists i)(i \leq n \wedge n = 2^i - 1)$ and defining the new relational constant $\leq n = \{\langle x, y\rangle : x, y \in Nat \wedge y \leq x\}$, it is easy to obtain the following equality

$$\left(I \otimes (pot2 ; pred)\right) ; (I \nabla I)^{\mathsf{T}} = (I \otimes \leq n) ; \left(I \otimes (pot2 ; pred)\right) ; (I \nabla I)^{\mathsf{T}}$$

Substituting we have

$$\ddot{I}_{prop} = I_{Nat} ; \pi^{\mathsf{T}} ; (I \nabla \leq n) ; \left(I \otimes (pot2 ; pred)\right) ; (I \nabla I)^{\mathsf{T}} ; true + I_{\overline{prop}} ; false$$

Which, by using (23) – Definition 11 –, we can write as

$$\ddot{I}_{prop} = I_{Nat} ; (I \nabla \infty) ; (I \otimes \leq n) ; \left(I \otimes (pot2 ; pred)\right) ; (I \nabla I)^{\mathsf{T}} ; true + I_{\overline{prop}} ; false$$

Notice that here we are in the crucial point of the derivation. Due to the fact that projections –and as a consequence cylindrifications– are not fundamental operations, we can write

$$(I \nabla \infty) ; (I \otimes \leq n) = (I \nabla \leq n)$$

thus restricting the *combinatorial explosion* produced by π^{T}. Then, we have

$$\ddot{I}_{prop} = I_{Nat} ; (I \nabla \leq n) ; \left(I \otimes (pot2 ; pred)\right) ; (I \nabla I)^{\mathsf{T}} ; true + I_{\overline{prop}} ; false$$

Distributing now ∇ over $\otimes$[13], we have

$$\ddot{I}_{prop} = I_{Nat} ; \left(I \nabla (\leq n ; pot2 ; pred)\right) ; (I \nabla I)^{\mathsf{T}} ; true + I_{\overline{prop}} ; false$$

But notice that $\leq n = pred^{i*}$, where $pred^{i*}$ is the transitive closure of $pred$. Then

$$\ddot{I}_{prop} = I_{Nat} ; \left(I \nabla (pred^{i*} ; pot2 ; pred)\right) ; (I \nabla I)^{\mathsf{T}} ; true + I_{\overline{prop}} ; false$$

which is an iterative algorithm.

5.4 Some Considerations About Residuals and Relational Implication

For the subsequent derivation, some discussion about residuals will be useful.

Let $s \backslash r$ be the *left residual* and s / r the *right residual*. Their relational expressions can be calculated by means of Galois connections from the equations $x ; r \subset s$ and $r ; x \subset s$, respectively, by using Schröder rule, i.e.,

$$x \subset s \backslash r \leftrightarrow x ; r \subset s \tag{27}$$

$$x \subset s / r \leftrightarrow r ; x \subset s \tag{28}$$

Since $s \backslash r$ and s / r are the greatest and the strongest solutions of equations $x ; r \subset s$ and $r ; x \subset s$, respectively, we can write $s \backslash r = \overline{s} ; r^{\mathsf{T}}$ and $s / r = r^{\mathsf{T}} ; \overline{s}$ (recall that $r \dagger s = \overline{\overline{r} ; \overline{s}}$).

Now, notice that

$$\left(s \backslash r\right)^{\mathsf{T}} = \left(\overline{\overline{s} ; r^{\mathsf{T}}}\right)^{\mathsf{T}} = \left(\overline{\overline{s} ; r^{\mathsf{T}}}\right)^{\mathsf{T}} = \overline{r ; (\overline{s})^{\mathsf{T}}} = \overline{r} \dagger s^{\mathsf{T}} \tag{29}$$

---

13 General properties of ∇ and $\otimes$, can be found in [Hae91].

then, by recalling the proper algebra definition of $r \dagger s$, we can write

$$(\forall x)(\forall y)\Big(x\big(s \not/ r\big)^{\mathsf{T}} y \leftrightarrow (\forall z)\big(x \bar{r} z \vee z s^{\mathsf{T}} y\big)\Big)$$

which can be rewritten as

$$(\forall x)(\forall y)\Big(x\big(s \not/ r\big)^{\mathsf{T}} y \leftrightarrow (\forall z)(x r z \rightarrow y s z)\Big)$$

This definition lead us to introduce a new symbol for the converse of the left residual –which in the sequel will be called *relational implication*–, namely, $r \rightarrowtail s = \big(s \not/ r\big)^{\mathsf{T}}$. Notice also that

$$r^{\mathsf{T}} \rightarrowtail s^{\mathsf{T}} = \big(s^{\mathsf{T}} \not/ r^{\mathsf{T}}\big)^{\mathsf{T}} = \left(\overline{\bar{s}^{\mathsf{T}} ; r^{\mathsf{T}\bar{\mathsf{T}}}}\right)^{\mathsf{T}} = \overline{r^{\mathsf{T}} ; \bar{s}} = s \not\backslash r \tag{30}$$

Then, $r^{\mathsf{T}} \rightarrowtail s^{\mathsf{T}} = \sup\{x : r ; x \subset s\} = \overline{r^{\mathsf{T}}} \dagger s$, thus

$$(\forall x)(\forall y)\Big(x r^{\mathsf{T}} \rightarrowtail s^{\mathsf{T}} y \leftrightarrow (\forall z)\big(z \overline{r^{\mathsf{T}}} x \vee z s y\big)\Big) \tag{31}$$

which can be rewritten as

$$(\forall x)(\forall y)\big(x r^{\mathsf{T}} \rightarrowtail s^{\mathsf{T}} y \leftrightarrow (\forall z)(z r x \rightarrow z s y)\big) \tag{32}$$

5.5 Specifications Involving Universal Quantifiers - Formal Development of a Constructive Specification of Quicksort from the Partial Identity over Ordered Lists

As was done in the preceding section, translation from first-order formulae to terms in ∇-Abstract Relational Algebra can be accomplished by the straightforward application of the constructive proof of the expressiveness Theorem 9. Here we will begin our formal development by doing another kind of translation, which we will call "by *pattern matching*".

Let ℓ, ℓ_1 and ℓ_2 be lists, $\mathcal{Ord}\ (\ell)$ a predicate whose truth-value is true iff ℓ is an ordered list and last and head the usual operations on lists. A first-order definition of $\mathcal{Ord}$ could be,

$$(\forall \ell)\Big(\mathcal{Ord}(\ell) \leftrightarrow (\forall \ell_1)(\forall \ell_2)\big(\ell \in \mathcal{L}^1 \vee \ell = \ell_1 \# \ell_2 \rightarrow \mathsf{last}(\ell_1) \leq \mathsf{head}(\ell_2)\big)\Big)$$

which can be written in Proper Relational Algebra as

$$(\forall \ell)\Big(\ell\ \mathcal{Ord}\ \mathsf{T} \leftrightarrow \big(\forall [\ell_1, \ell_2]\big)\big(\ell \in \mathcal{L}^1 \vee [\ell_1, \ell_2]\ cnc\ \ell \rightarrow [\ell_1, \ell_2]\ (lst \otimes hd) ; 1_{\leq}\big)\ \mathrm{pair}\Big) \tag{33}$$

where,

- $\mathcal{L}^1$ is the set of lists of length 1,
- $\mathcal{L}^*$ is the set of all lists,
- $1_{\mathcal{L}^*}$ is the identity relation restricted to the set of all lists,
- $1_{\mathcal{L}^i}$ is the identity restricted to the set of lists of length i or less,
- *lst* is a relation which for each list gives its last element,
- *hd* is a relation giving for each list its head,
- *cnc* is a relation which given two lists of length 1 or greater concatenates them to form a new list[14] and
- $1_{\leq}$ is the partial identity over pairs pair of elements satisfying relation $\leq$.

---

14 Notice that *cnc* is a partial relation not defined on empty lists.

Comparing expressions (33) and (32) (see figure 6) it is evident that both definiens match. Then, replacing them by its definienda we can write

$$(\forall \ell)\left(\ell \, Ord \; \mathsf{T} \leftrightarrow 1_{\mathcal{L}^1} + \ell\left(cnc^{\mathsf{T}} \rightarrow \left((lst \otimes hd); 1_{\leq}\right)^{\mathsf{T}}\right)\text{pair}\right) \tag{34}$$

recalling that $1 \bullet (r; \infty)$ is the identity over the domain of r, we can write

$$1_{Ord} = 1_{\mathcal{L}^*} \bullet (Ord; \infty) = 1_{\mathcal{L}^*} \bullet \left(1_{\mathcal{L}^1} + \left(cnc^{\mathsf{T}} \rightarrow \left((lst \otimes hd); 1_{\leq}\right)^{\mathsf{T}}\right); \infty\right)$$

distributing intersection over sum, we have .

$$1_{Ord} = \underbrace{1_{\mathcal{L}^*} \bullet 1_{\mathcal{L}^1}}_{1_{\mathcal{L}^1}} + 1_{\mathcal{L}^*} \bullet \left(\left(cnc^{\mathsf{T}} \rightarrow \left((lst \otimes hd); 1_{\leq}\right)^{\mathsf{T}}\right); \infty\right) \tag{35}$$

Notice that the expression $\left(\left(cnc^{\mathsf{T}} \rightarrow \left((lst \otimes hd); 1_{\leq}\right)^{\mathsf{T}}\right); \infty\right)$ is not defined over $\mathcal{L}^1$.

Then, assigning to operation $+$ angelic non determinism [Ber86], we can write the following deterministic expression

$$1_{Ord} = 1_{\mathcal{L}^1} + 1_{\mathcal{L}^* - \mathcal{L}^1} \bullet \left(\left(cnc^{\mathsf{T}} \rightarrow \left((lst \otimes hd); 1_{\leq}\right)^{\mathsf{T}}\right); \infty\right) \tag{36}$$

$$\overbrace{(\forall \ell)}^{(\forall x)(\forall y)} \quad (\ell \, Ord \; \mathsf{T} \leftrightarrow \overbrace{(\forall [\ell_1, \ell_2])}^{(\forall z)} (\overbrace{\ell \in \mathcal{L}^1 \vee [\ell_1, \ell_2] \; conc \; \ell}^{z \, r \, x} \rightarrow \overbrace{[\ell_1, \ell_2].((lst \otimes hd); 1) \; \text{pair})}^{z \, s \, y})$$

$$(x \, r^{\mathsf{T}} \rightarrow s^{\mathsf{T}} \, y \leftrightarrow (\forall z) \quad (\quad z \, r \, x \quad \rightarrow \quad z \, s \, y \quad)$$

Fig. 6. Matching patterns between first-order predicate *Ord* and Proper Relational Algebra definition of relational implication between converses

Let us forget the trivial part $1_{\mathcal{L}^1}$ and concentrate ourselves in the second non trivial part of (36). Thus, calling $1_{Ord\,2} = 1_{Ord} - 1_{\mathcal{L}^1}$ and recalling (30) we can introduce the right residual

$$1_{Ord\,2} = 1_{\mathcal{L}^* - \mathcal{L}^1} \bullet \left(\left((lst \otimes hd); 1_{\leq} \lfloor cnc\right); \infty\right) \tag{37}$$

which is equivalent to

$$1_{Ord\,2} = 1_{\mathcal{L}^* - \mathcal{L}^1} \bullet \left(\sup\{x : cnc; x \subset (lst \otimes hd); 1_{\leq}\}; \infty\right) \tag{38}$$

Notice that we must solve the equation $\sup\{x : cnc; x \subset (lst \otimes hd); 1_{\leq}\}$ over the set of ordered lists. For doing so, we can introduce the following two lemmas:

Lemma 31 If $Dom(x)$ is the set of lists ℓ satisfying $\ell \, Ord \; \mathsf{T}$ then $cnc; x = \left(1_{Ord} \otimes 1_{Ord}\right); cnc; x$.

Proof $(x; \infty) \bullet 1 = 1_{Ord} \rightarrow \left(1_{Ord} \otimes 1_{Ord}\right); cnc; x = \left(1_{Ord} \otimes 1_{Ord}\right); cnc; 1_{Ord}; x$

but $\left(1_{Ord} \otimes 1_{Ord}\right); cnc; 1_{Ord} = \left(1_{\mathcal{L}^*} \otimes 1_{\mathcal{L}^*}\right); cnc; 1_{Ord}$

then $\left(1_{Ord} \otimes 1_{Ord}\right); cnc; x = cnc; 1_{Ord}; x = cnc; x$

Q. E. D.

Lemma 32 $\sup\{x : 1_{Ord} = 1 \bullet (x; \infty) \wedge cnc; x \subset (lst \otimes hd); 1_{\leq}\} =$
$cnc^{\mathsf{T}}; \left(1_{Ord} \otimes 1_{Ord}\right); (lst \otimes hd); 1_{\leq}$

Proof $\{x : 1_{Ord} = 1 \bullet (x;\infty) \wedge cnc;\ x \subset (lst \otimes hd);\ 1_{\leq}\} =$

$= \{x : 1_{Ord} = 1 \bullet (x;\infty) \wedge (1_{Ord} \otimes 1_{Ord});\ cnc;\ x \subset$

$(1_{Ord} \otimes 1_{Ord});\ (lst \otimes hd);\ 1_{\leq}\}$

by Lemma 31 this set equals

$$\{x : 1_{Ord} = 1 \bullet (x;\infty) \wedge cnc;\ x \subset (1_{Ord} \otimes 1_{Ord});\ (lst \otimes hd);\ 1_{\leq}\}$$

by ; monotonicity and since *cnc* is univalent we can write the above set as

$$\{x : 1_{Ord} = 1 \bullet (x;\infty) \wedge x \subset cnc^{\mathsf{T}};\ (1_{Ord} \otimes 1_{Ord});\ (lst \otimes hd);\ 1_{\leq}\}$$

then $\sup\{x : 1_{Ord} = 1 \bullet (x;\infty) \wedge cnc;\ x \subset (lst \otimes hd);\ 1_{\leq}\} =$
$cnc^{\mathsf{T}};\ (1_{Ord} \otimes 1_{Ord});\ (lst \otimes hd);\ 1_{\leq}$

so, $cnc^{\mathsf{T}};\ (1_{Ord} \otimes 1_{Ord});\ (lst \otimes hd);\ 1_{\leq}$ is an upper bound and since its domain equals the one of Ord

Q. E. D.

Then, we can rewrite (38) as

$$1_{Ord\,2} = 1_{\mathcal{L}^* - \mathcal{L}^1} \bullet \left(\left(cnc^{\mathsf{T}};\ (1_{Ord} \otimes 1_{Ord});\ (lst \otimes hd);\ 1_{\leq}\right);\infty\right)$$

which obviously equals

$$1_{Ord\,2} = 1_{Ord} \bullet \left(\left(cnc^{\mathsf{T}};\ (1_{Ord} \otimes 1_{Ord});\ (lst \otimes hd);\ 1_{\leq}\right);\infty\right)$$

Now, recalling Proposition 21 we can write the above expression as

$$1_{Ord\,2} = \begin{pmatrix} & & \\ & 1_{Ord} & \\ & \nabla & \\ cnc^{\mathsf{T}};\ \begin{pmatrix} 1_{Ord} \\ \otimes \\ 1_{Ord} \end{pmatrix};\ \begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix};\ 1_{\leq} & & \end{pmatrix};\ \pi$$

which obviously equals

$$1_{Ord\,2} = \begin{pmatrix} 1_{Ord};\ cnc^{\mathsf{T}};\ \begin{pmatrix} 1_{Ord} \\ \otimes \\ 1_{Ord} \end{pmatrix};\ cnc & & \\ & \nabla & \\ cnc^{\mathsf{T}};\ \begin{pmatrix} 1_{Ord} \\ \otimes \\ 1_{Ord} \end{pmatrix};\ \begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix};\ 1_{\leq} & & \end{pmatrix};\ \pi$$

But, by Lemma 32, the domain of $cnc^{\mathsf{T}};\ (1_{Ord} \otimes 1_{Ord});\ (lst \otimes hd);\ 1_{\leq}$ is the set of ordered lists. Then, from the definition of ∇ we have

$$1_{Ord\,2} = \left(\begin{array}{c} cnc^{\mathsf{T}} ; \begin{pmatrix} 1_{Ord} \\ \otimes \\ 1_{Ord} \end{pmatrix} ; cnc \\ \nabla \\ cnc^{\mathsf{T}} ; \begin{pmatrix} 1_{Ord} \\ \otimes \\ 1_{Ord} \end{pmatrix} ; \begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix} ; 1_{\leq} \end{array} \right) ; \pi$$

But from Propositions 26 and 28 and Corollaries 29 and 30 and Definition 25 we can easily derive $1 \circ (r ; s ; \infty) = 1 \circ (r ; t ; \infty) \rightarrow r ; \begin{pmatrix} s \\ \nabla \\ t \end{pmatrix} \sqsubseteq \begin{pmatrix} r ; s \\ \nabla \\ r ; t \end{pmatrix}$, and since 1_{Ord} is univalent, $\sqsubseteq$ is equivalent to $=$. Then, we can write

$$1_{Ord\,2} = cnc^{\mathsf{T}} ; \left(\begin{array}{c} \begin{pmatrix} 1_{Ord} \\ \otimes \\ 1_{Ord} \end{pmatrix} ; cnc \\ \nabla \\ \begin{pmatrix} 1_{Ord} \\ \otimes \\ 1_{Ord} \end{pmatrix} ; \begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix} ; 1_{\leq} \end{array} \right) ; \pi$$

and then,

$$1_{Ord\,2} = cnc^{\mathsf{T}} ; \left(\begin{array}{c} \begin{pmatrix} 1_{Ord} \\ \otimes \\ 1_{Ord} \end{pmatrix} ; cnc \\ \nabla \\ \begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix} ; 1_{\leq} \end{array} \right) ; \pi \qquad (39)$$

It is important to notice that what we have carried out was the construction of a recursive refinement (39) for the universal quantified relational expression $1_{L^* - L^1} \bullet \left(\left(cnc^{\mathsf{T}} \rightarrowtail \left((lst \otimes hd) ; 1_{\leq} \right)^{\mathsf{T}} \right) ; \infty \right)$. In doing so, we could resort to the Galois connection,

$$x \subset \left((lst \otimes hd) ; 1_{\leq} \lfloor cnc \right) \leftrightarrow cnc ; x \subset (lst \otimes hd) ; 1_{\leq}$$

However. the intuition needed for stating equation $cnc ; x \subset (lst \otimes hd) ; 1_{\leq}$ is not trivial. Notice also that the solution on x of this equation must be a relation with the same domain as Ord. Then, to guaranteeing the inclusion

$x \subset cnc^{\mathsf{T}} ; (1_{Ord} \otimes 1_{Ord}) ; (lst \otimes hd) ; 1_{\leq}$ the domain of x must be the domain of ordered lists. What we has done is to replace "clever intuition" by mere calculation.

Let us now continue with the calculation of our sorting algorithm. An obvious specification for *sort* is,

$$sort = perm ; 1_{Ord} \tag{40}$$

which by trivialising and distributing it, and recalling that $perm ; 1_{L^1} = 1_{L^1}$, we can write

$$sort = perm ; (1_{L^1} + 1_{Ord\,2}) = perm ; 1_{L^1} + perm ; 1_{Ord\,2} = 1_{L^1} + perm ; 1_{Ord\,2}$$

By unfolding (39) on it we have

$$sort = 1_{L^1} + 1_{L^* - L^1} ; perm ; cnc^{\mathsf{T}} ; \left(\begin{pmatrix} 1_{Ord} \\ \otimes \\ 1_{Ord} \end{pmatrix} ; cnc \;\; \nabla \;\; \begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix} ; 1_{\leq} \right) ; \pi$$

By applying now Proposition 27 we obtain

$$sort = 1_{L^1} + 1_{L^* - L^1} ; perm ; cnc^{\mathsf{T}} ; \left(\left(\begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix} ; 1_{\leq} ; \infty \right) \circ 1 \right) ; \begin{pmatrix} 1_{Ord} \\ \otimes \\ 1_{Ord} \end{pmatrix} ; cnc$$

which by commutativity of identities we can rewrite as

$$sort = 1_{L^1} + 1_{L^* - L^1} ; perm ; cnc^{\mathsf{T}} ; \begin{pmatrix} 1_{Ord} \\ \otimes \\ 1_{Ord} \end{pmatrix} ; \left(\left(\begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix} ; 1_{\leq} ; \infty \right) \circ 1 \right) ; cnc$$

But $perm ; cnc^{\mathsf{T}} = perm ; cnc^{\mathsf{T}} ; \begin{pmatrix} perm \\ \otimes \\ perm \end{pmatrix}$. Substituting the right hand side in the last expression of *sort* we have

$$sort = 1_{L^1} + 1_{L^* - L^1} ; perm ; cnc^{\mathsf{T}} ; \begin{pmatrix} perm \\ \otimes \\ perm \end{pmatrix} ; \begin{pmatrix} 1_{Ord} \\ \otimes \\ 1_{Ord} \end{pmatrix} ; \left(\left(\begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix} ; 1_{\leq} ; \infty \right) \circ 1 \right) ; cnc$$

Applying distributivity of $\otimes$ with respect to ;, we have

$$sort = 1_{L^1} + 1_{L^* - L^1} ; perm ; cnc^{\mathsf{T}} ; \begin{pmatrix} perm ; 1_{Ord} \\ \otimes \\ perm ; 1_{Ord} \end{pmatrix} ; \left(\left(\begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix} ; 1_{\leq} ; \infty \right) \circ 1 \right) ; cnc$$

By unfolding (40) here, we have

$$sort = 1_{\mathcal{L}^1} + 1_{\mathcal{L}^\bullet - \mathcal{L}^1} \,;\, perm \,;\, cnc^{\mathsf{T}} \,;\, \begin{pmatrix} sort \\ \otimes \\ sort \end{pmatrix} \,;\, \left(\left(\begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix} \,;\, 1_{\leq} \,;\, \infty \right) \bullet 1 \right) \,;\, cnc$$

Rewriting $\left(\begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix} \,;\, 1_{\leq} \,;\, \infty \right) \bullet 1$ using fork

$$sort = 1_{\mathcal{L}^1} + 1_{\mathcal{L}^\bullet - \mathcal{L}^1} \,;\, perm \,;\, cnc^{\mathsf{T}} \,;\, \begin{pmatrix} sort \\ \otimes \\ sort \end{pmatrix} \,;\, \begin{pmatrix} \begin{pmatrix} 1_{\mathcal{L}^\bullet} \\ \otimes \\ 1_{\mathcal{L}^\bullet} \end{pmatrix} \\ \nabla \\ \begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix} \,;\, 1_{\leq} \end{pmatrix} \,;\, \pi \,;\, cnc$$

Since *sort* is univalent we can distribute ; over ∇, and noticing that $\begin{pmatrix} sort \\ \otimes \\ sort \end{pmatrix} \,;\, \begin{pmatrix} 1_{\mathcal{L}^\bullet} \\ \otimes \\ 1_{\mathcal{L}^\bullet} \end{pmatrix} = \begin{pmatrix} sort \\ \otimes \\ sort \end{pmatrix}$, we obtain

$$sort = 1_{\mathcal{L}^1} + 1_{\mathcal{L}^\bullet - \mathcal{L}^1} \,;\, perm \,;\, cnc^{\mathsf{T}} \,;\, \begin{pmatrix} \begin{pmatrix} sort \\ \otimes \\ sort \end{pmatrix} \\ \nabla \\ \begin{pmatrix} sort \\ \otimes \\ sort \end{pmatrix} \,;\, \begin{pmatrix} lst \\ \otimes \\ hd \end{pmatrix} \,;\, 1_{\leq} \end{pmatrix} \,;\, \pi \,;\, cnc$$

By distributing $\otimes$ over ; in the inferior branch of the fork

$$sort = 1_{\mathcal{L}^1} + 1_{\mathcal{L}^\bullet - \mathcal{L}^1} \,;\, perm \,;\, cnc^{\mathsf{T}} \,;\, \begin{pmatrix} \begin{pmatrix} sort \\ \otimes \\ sort \end{pmatrix} \\ \nabla \\ \begin{pmatrix} sort \,;\, lst \\ \otimes \\ sort \,;\, hd \end{pmatrix} \,;\, 1_{\leq} \end{pmatrix} \,;\, \pi \,;\, cnc$$

But noticing that $sort \,;\, lst = max$ and $sort \,;\, hd = min$ and folding, we obtain

$$sort = 1_{\mathcal{L}^1} + 1_{\mathcal{L}^* - \mathcal{L}^1} ; perm ; cnc^{\mathsf{T}} ; \left(\begin{array}{c} \begin{pmatrix} sort \\ \otimes \\ sort \end{pmatrix} \\ \nabla \\ \begin{pmatrix} max \\ \otimes \\ min \end{pmatrix} ; 1_{\leq} \end{array} \right) ; \pi ; cnc$$

Finally, by applying Proposition 27 we obtain

$$sort = 1_{\mathcal{L}^1} + 1_{\mathcal{L}^* - \mathcal{L}^1} ; perm ; cnc^{\mathsf{T}} ; \left(\begin{array}{c} 1 \\ \nabla \\ \begin{pmatrix} max \\ \otimes \\ min \end{pmatrix} ; 1_{\leq} \end{array} \right) ; \pi ; \begin{pmatrix} sort \\ \otimes \\ sort \end{pmatrix} ; cnc \quad (41)$$

Notice that if we do $partition = perm ; cnc^{\mathsf{T}} ; \left(\begin{array}{c} 1 \\ \nabla \\ \begin{pmatrix} max \\ \otimes \\ min \end{pmatrix} ; 1_{\leq} \end{array} \right) ; \pi$ and fold (41) we have

$$sort = 1_{\mathcal{L}^1} + 1_{\mathcal{L}^* - \mathcal{L}^1} ; partition ; \begin{pmatrix} sort \\ \otimes \\ sort \end{pmatrix} ; cnc$$

which is obviously a specification of Quicksort. A sort algorithm can be derived by calculating out of the specification of *partition*..

6 Conclusions

We have presented an alternative basis for relational programming calculi by extending Abstract Relational Algebras with a fork operator. This operator, as well as the direct product and the projections, has being used by other researchers. However, they fail to formulate that the new algebra has the expressive power of first-order logic[15]. We introduce an axiomatization for the ∇-Extended Abstract Relational

15 Notice that as late as 1991 Maddux [Mad91] and Németi [Ném91] considered this problem unsolved.

Algebra, compare it to "classical" algebras for first-order logic and, later, proved that alternative axiomatizations for fork and projections can be derived from our one.
At this point we show, by formally calculating algorithms out of first-order specifications of two "classical" examples, the advantage of having quantifiers expressed by terms. We also show two ways of translating first-order expressions onto relational terms. Two main points of this second part of the paper are the examples of how to cope with the algorithmic complexity of the relational term expressing existential quantifier and the proposed heuristic for deriving the equation on residuals expressing universal quantifier.
A point to be emphasised is that we will have another indication of the fitness of this algebra to underlay relational programming calculi, if it turns out by subsequent research that the ∇-Extended Relational Algebra is representable without the inclusion of the extra axiom (26).
Future research will be carried out along two main directions. The first one is the exhaustive analysis of the ∇-Extended Relational Algebra as an algebraic structure. The second one is the full development of a programming calculus. This involves answering the question of whether or not we should deal not trivially with partiality (as discussed in Section 4 of [Hae91]). After a conclusive answer to this question, we will develop a typing system for our evolving relational programming calculus.

References

[Bac92] Backhouse, R. and Tormenta, P., "Polynomial Relators" in *State-of-the-Art Seminar on Formal Program Development,* IFIP - WG 2.1 Algorithmic Languages and Calculi, 1992.

[Bau92] Baum, G., Haeberer, A.M., and Veloso, P.A.S., "On the Representability of the ∇-Abstract Relational Algebra" *IGPL Newsletter*, vol. 1, no. 3, , European Foundation for Logic, Language and Information Interest Group on Programming Logic.

[Bed78] Bednarek, A. R. and Ulam, S. M., "Projective Algebra and the Calculus of Relations" *Journal of Symbolic Logic*, vol. 43, no. 1, 56 - 64, March 1978.

[Ber86] Berghammer, R. and Zierer, H., "Relational Algebraic Semantics of Deterministic and Non-deterministic Programs" *Theoretical Computer Science*, vol. 43, 123 - 147, 1986, North-Holland.

[Ber91] Berghammer, R., "Relational Specification of Data Types and Programs" Tech. Rep., , , September 1991.

[Ber93] Berghammer, R., Haeberer, A.M., Schmidt, G., and Veloso, P.A.S., "Comparing two Different Approaches to Products in Abstract Relation Algebras" December 1992, Submitted to the Third International Conference on Algebraic Methodology and Software Technology, AMAST'93.

[Chi48] Chin, L. and Tarski, A., "Remarks on Projective Algebras" *Bulletin of the American Mathematical Society* , vol. 54, 80 - 81, 1948.

[Chi51] Chin, L. H. C. and Tarski, A., "Distributive and Modular Laws in the Arithmetic of Relation Algebras" in *University of California Publications in Mathematics*, of California, U., Ed. University of California, 1951, pp. 341 - 384.

[Ebb80] Ebbinghaus, H. D., Flum, J., and Thomas, W., *Mathematical Logic*. New York: Spriger-Verlag, 1980.

[Eve46] Everett, C. J. and Ulam, S. M., "Projective Algebra," *American Journal of Mathematics*, vol. 68, 77 - 88, 1946.

[Fri92] Frias, M. and Wachenchauzer, R., "Haeberer-Veloso's Relational Algebra as a Language to Express and Optimise Queries to a Relational Data Base" in *44th Meeting of the,* vol. Document IFIP Working Group 2.1 Algorithmic Languages and Calculi, 1992.

[Hae90] Haeberer, A.M., Veloso, P.A.S., and Elustondo, P., "Towards a Relational Calculus for Software Construction" in *41st Meeting of the,* vol. Document IFIP Working Group 2.1 Algorithmic Languages and Calculi, Chester - England, 1990.

[Hae91] Haeberer, A.M. and Veloso, P.A.S., "Partial Relations for Program Derivation: Adequacy, Inevitability and Expressiveness" in *Constructing Programs From Specifications - Proceedings of the IFIP TC2 Working Conference on Constructing Programs From Specifications,* N. H., IFIP WG 2.1, Bernhard Möller, 1991, pp. 319 - 371.

[Hal62] Halmos, P. R., *Algebraic Logic*. New York: Chelsea Publishing Company, 1962.

[Hen74] Henkin, L. and Monk, J. D., "Cylindric Algebras and Related Structures" in *Proceedings of the Tarski Symposium,* vol. 25 American Mathematical Society, 1974, pp. 105 - 121.

[Hoa86a] Hoare, C. A. R. and He, J., "The Weakest Presepecification, Part I" *Fundamenta Informatica*, vol. 4, no. 9, 51 - 54, 1986.

[Hoa86b] Hoare, C. A. R. and He, J., "The Weakest Presepecification, Part I" *Fundamenta Informatica*, vol. 4, no. 9, 217 - 252, 1986.

[Jón51] Jónsson, B. and Tarski, A., "Boolean Algebras With Operators PART I" *American Journal of Mathematics*, vol. 73, 891 - 939, 1951.

[Jón52] Jónsson, B. and Tarski, A., "Boolean Algebras With Operators PART II" *American Journal of Mathematics*, vol. 74, 127 - 162, 1952.

[Lyn50] Lyndom, R., "The Representation of Relational Algebras" *Annals of Mathematics (series 2)*, vol. 51, 707 - 729, 1950.

[Mad83] Maddux, R., "A Sequent Calculus for Relation Algebras" *Annals of Pure and Apply Logic*, vol. 25, 73 - 101, 1983.

[Mad91] Maddux, R., "The Origin of Relation Algebras in the Development and Axiomatization of the Calculus of Relations" *Studia Logica*, vol. L, no. 3 - 4, 412 - 455, 1991.

[Ném91] Németi, I., "Algebraization of Quantifier Logics, an Introductory Overview" *Studia Logica*, vol. L, no. 3-4, 485 - 569, 1991.

[Roe72] Roever, W. P. de, "A Formalization of Various Prameter Mechanisms as Products of Relations Within a Calculus of Recursive Program Schemes" in *Theorie des Algorithmes, des Languages et de la Programation*, 1972, pp. 55 - 88.

[Sch85a] Schmidt, G. and Ströhlein, T., "Relation Algebras: Concept of Points and Representability" *Discrete Mathematics*, vol. 54, 83 - 92, 1985.

[Sch85b] Schmidt, G. and Ströhlein, T., "Diskrete Mathematik–Relationen, Graphen und Programme" Report TU München.

[Sch89] Schmidt, G. and Ströhlein, T., *Relationen und Graphen*. Mathematik für Informatiker, Springer-Verlag, 1989.

[Sch93] Schmidt, G. and Ströhlein, T., *Relations and Graphs, Discret Mathematics for Computer Science*. EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1993.

[Tar41] Tarski, A., "On the Calculus of Relations" *Journal of Symbolic Logic*, vol. 6, 73 - 89, 1941.

[Tar52] Tarski, A. and Thompson, F. B., "Some General Properties of Cylindric Algebras" *Bulletin of the American Mathematical Society*, vol. 58, 65, 1952.

[Vel91a] Veloso, P.A.S. and Haeberer, A.M., "A New Algebra of First-Order Logic" in *9th International Congress on Logic, Methodology and Philosophy of Science*, Upsala, Sweden, 1991.

[Vel91b] Veloso, P.A.S. and Haeberer, A.M., "A Finitary Relational Algebra For Classical First-Order Logic" *Bulletin of the Section of Logic of the Polish Academy of Sciences*, vol. 20, no. 2, 52 - 62, June 1991.

[Vel92] Veloso, P.A.S., Haeberer, A.M., and Baum, G., "Formal Program Construction Within an Extended Calculus of Binary Relations" Tech. Rep., Res. Rept. , , , MCC 19, 1992, Submitted to an special issue on Automatic Programming of the Journal of Symbolic Computation.

[Zie83] Zierer, H., "Relationale Semantik" Ph.D. thesis, Institut für Informatik, Technische Universität München, 1983.