NORMALISATION OF DATABASE EXPRESSIONS INVOLVING CALCULATIONS

Sieger van Denneheuvel
Logic and Computation Theory Group, Department of Mathematics and Computer Science, University of Amsterdam

Gerard R. Renardel de Lavalette
Logic Group, Department of Philosophy, University of Utrecht

Logic Group

Preprint Series

No. 45

Department of Philosophy

University of Utrecht

NORMALISATION OF DATABASE EXPRESSIONS INVOLVING CALCULATIONS

Sieger van Denneheuvel
Logic and Computation Theory group, Department of Mathematics and Computer Science, University of Amsterdam

Gerard R. Renardel de Lavalette
Logic Group, Department of Philosophy, University of Utrecht

Abstract

In this paper we introduce a relational algebra extended with a *calculate* operator and derive, for expressions in the corresponding language **PCSJL**, a normalisation procedure. **PCSJL** plays a role in the implementation of the Rule Language **RL**; the normalisation is to be used for query optimisation.

Keywords

relational data base, relational algebra, query optimisation, normalisation

Logic Group Preprint Series nr. 45
Department of Philosophy
University of Utrecht
Heidelberglaan 2
3584 CS Utrecht
the Netherlands

March 1989

1. Introduction

PSJ expressions are relational algebra expressions containing only project, select and join operators. This restricted class of expressions is commonly used in relational databases. PSJ expressions are studied in [YAN87] and [LAR85], where it is mentioned without proof (which is not very difficult) that they can be transformed into a normal form where first the join operators are applied, then selection and finally projection. Such normalisation procedures play an important role in query optimisation: see [ULL82, Ch. 8], [YAN87]. Standard optimisation techniques can be used to further optimize PSJ normal form expressions: e.g. the 'selection before join' heuristic can be applied to push selection down to the relational database tables ([ULL82]).

In this paper we add the relational operator *calculate* to the above mentioned relational operators, thus obtaining the language **PCSJL** of PCSJ expressions. The question arises whether PCSJ expressions also can be brought into a normal form, where the joins are followed by selection, then calculate and finally projection. It is shown that such a normal form exists (§4); moreover, the proof yields a construction that has been converted into an implemented algorithm (§5). This PCSJ normal form procedure already provides optimisation, and the normal forms it yields can serve as the starting point for further optimisation (just as for PSJ normal forms).

Our interest for PCSJ expressions lies in its role in the integration of relational databases and constraint solving. This integration is one of the aims of the declarative Rule Language **RL**. The potential for such an integration has been investigated in the context of the Rules Technology project led by Peter Lucas at the IBM San Jose Research Center: see e.g. [HANS87],[HANS88]. **RL** was defined by Peter van Emde Boas in [VEMD86a], where a relational semantic model is given to interpret **RL** (see also [VEMD86b], [VEMD86c]). A considerable part of this language has been implemented by the first author of this paper: see [DEN88].

RL can be considered as an extension of **SQL** with existential quantification over variables occurring in constraints but not necessarily in relations (as is required in **SQL**, where all variables in the WHERE clause have to be present in the FROM clause; see [DATE87], [DATE88]). As a consequence, not all expressions in **RL** can be evaluated: imagine what happens when the existential quantifier ranges over an infinite domain. To be able to deal with these problems, the above-mentioned implementation of **RL** is equipped with a *constraint solver*. This constraint solver transforms evaluable **RL**-expressions into expressions of **PCSJL**, which corresponds to the fragment of **RL**

without existential quantification.

Outline of the rest of this paper. In §2 we introduce the constraint language **CL** which serves as a parameter of the language **PCSJL**, defined in §3; §4 contains the normalisation procedure, in §5 we shortly mention its implementation, finally in §6 we present conclusions and a perspective on future research.

Acknowledgements. The authors thank Peter van Emde Boas, Karen Kwast and Edith Spaan (all University of Amsterdam) for useful criticism and remarks.

2. The constraint language CL

We begin with the definition of the *constraint language* **CL**: it will act as a parameter of the language **PCSJL** defined in §3. **CL** is a many-sorted language containing variables (denoted by the metavariables x,y,..., also called *attributes*), constants (c,d,..., also called *values*), functions (f,g,...), = (the equality predicate), predicates, propositional connectives ($\neg,\wedge,\vee,\rightarrow$) and the propositional constant **true**. Terms (s,t,...) and assertions (A,B,..., also called *constraints* or *conditions*) are defined as usual.

If I is any of the items defined here (or a collection of these), then var(I) is the collection of variables occurring in I.

Furthermore, we assume some evaluation mechanism $|\,.\,|$ for **CL** to be given, which evaluates closed terms (terms without variables) to constants and closed assertions to truth values.

We give an example language for **CL**, defined by the following sorts, constants, functions and predicates:

sorts: NUM (natural numbers), STR (strings of characters)

constants: 0,1,2,... in NUM; all finite strings in STR

functions: $*, + : \text{NUM} \times \text{NUM} \rightarrow \text{NUM}$
$\| : \text{STR} \times \text{STR} \rightarrow \text{STR}$ (concatenation)
length: $\text{STR} \rightarrow \text{NUM}$ (length of a string)
digits: $\text{NUM} \rightarrow \text{STR}$ (converts a number to its string representation)

predicates: $<, >, \leq, \geq, \neq$ (binary predicates, both on NUM and on STR)

3. The language PCSJL

3.1. Before we define the sorts of the language **PCSJL**, we introduce the following. *Solutions* are constraints of the form $x=t$ with $x \notin var(t)$. A *solution set* is a finite set $\{x_1=t_1,...,x_n=t_n\}$, satisfying

$\|\{x_1,...,x_n\}\| = n$, i.e. the head variables are all distinct;

$\{x_1,...,x_n\} \cap var(\{t_1,...,t_n\}) = \emptyset$.

A *tuple* is a solution set of the form $\{x_1=c_1,...,x_n=c_n\}$. For tuples Θ, we often write $attr(\Theta)$ instead of $var(\Theta)$. Tuples are called *similar* if they have the same attributes. A *relation* R is a pair $<X,R'>$ of a finite collection of attributes X and a finite collection of similar tuples, satisfying

$\forall \Theta \in R'$ $attr(\Theta)=X$.

If R' is nonempty then X can be obtained from R'; since most relations are nonempty, we shall allow ourselves to be a bit sloppy and identify R and R', i.e. consider a base relation to be a collection of similar tuples.

3.2. Assume that an instance of **CL** is given, i.e. some language with sorts, variables, constants, etc. We now present the definition of the language **PCSJL** = **PCSJL(CL)**; the interpretation of the language is given together with its definition. (Methodologically, it might be more correct to separate syntax and semantics, but we think this would not yield an easier understanding; moreover, it will always be clear what is syntax and what is semantics.)

PCSJL is a four-sorted language with expressions (thus named to distinguish them from **CL**-terms) and equations. The sorts are

VAR (finite sets of **CL**-variables)

CON (constraints, i.e. **CL**-assertions)

SOL (solution sets)

REL (relations)

We let X,Y,Z range over VAR; A,B,C over CON; Φ,Ψ over SOL; R,S over REL.

3.3. In this section, we present the functions of **PCSJL**. They are grouped according to their range.

3.3.1. Functions with range VAR.

Besides the usual set operations $\cup$, $\cap$ and $-$, we have

attr: REL → VAR

var: CON → VAR

hvar: SOL → VAR

tvar: SOL → VAR

the latter two defined by

hvar($\{x_1=t_1,...,x_n=t_n\}$) = $\{x_1,...,x_n\}$, the *head* variables, and

tvar($\{x_1=t_1,...,x_n=t_n\}$) = var($\{t_1,...,t_n\}$), the *tail* variables of a solution set.

3.3.2. Functions with range CON.

Besides $\wedge$ (conjunction), we have the *merge* of two solution sets, yielding a constraint. This is defined as follows:

$\_\oplus\_$: SOL × SOL → CON

$\Phi \oplus \Psi$ = (the conjunction of $\{s=t \mid x=s \in \Phi, x=t \in \Psi\}$)

Solution sets $\Phi = \{x_1=t_1,...,x_n=t_n\}$ can be interpreted as substitutions $[x_1:=t_1,...,x_n:=t_n]$ which can be applied to (collections of) items. So we have an operation *apply*:

$\_(\_)$: SOL × CON → CON

$\Phi(A)$ = (Φ, considered as a substitution, applied to A)

Furthermore we have the function ρ (restrict):

ρ: SOL × VAR → CON

$\rho(\Phi,X)$ = (the conjunction of $\{x=t \in \Phi \mid x \in X\}$)

3.3.3. Functions with range SOL.

Here, too, we have the usual set operations $\cup$, $\cap$ and $-$; besides, we introduce the function δ (delete):

δ: SOL × VAR → SOL

$\delta(\Phi,X) = \{x=t \in \Phi \mid x \notin X\}$

We overload the operation *apply*:

$\_(\_)$: SOL × SOL → SOL

$\Phi(\Psi) = \{x=\Phi(t) \mid x=t \in \Psi)$

3.3.4. Functions with range REL.

Here we find the usual operations π (projection), σ (selection) and $\otimes$ (join) on relations, together with κ (calculate). The definitions are

π: REL $\times$ VAR $\rightarrow$ REL

σ: REL $\times$ CON $\rightarrow$ REL

κ: REL $\times$ SOL $\rightarrow$ REL

$\_\otimes\_$: REL $\times$ REL $\rightarrow$ REL

$\pi(R,X) = \{\rho(\Theta,X) \mid \Theta \in R\}$ provided $X \subseteq attr(R)$

$\sigma(R,A) = \{\Theta \in R \mid |\Theta(A)| = \textbf{true}\}$ provided $var(A) \subseteq attr(R)$

$\kappa(R,\Phi) = \{\Theta \cup \Theta(\Phi) \mid \Theta \in R\}$ provided $tvar(\Phi) \subseteq attr(R)$ and $hvar(\Phi) \cap attr(R) = \emptyset$

$R \otimes S = \{\Theta \cup \Theta' \mid \Theta \in R, \Theta' \in S, \forall x \in attr(\Theta) \cap attr(\Theta') (\Theta(x) = \Theta'(x))\}$

One readily observes that the functions π, σ and κ are *partial*, i.e. they are only defined when certain conditions on the arguments are met, these conditions are referred to as the *definedness* conditions. They are quite reasonable: the definedness condition for projection ensures that a relation is not projected on attributes that are not part of the relation; the definedness condition for selection takes care that the constraint A can indeed be evaluated to **true** or **false**; the first part of the definedness condition for the calculate operator ensures that the tails of solutions in Φ can be evaluated, the second part rules out the possibility that the head of a solution is also determined directly by an attribute of the relation R.

The functions defined in §3 can be represented as in figure 1 below.

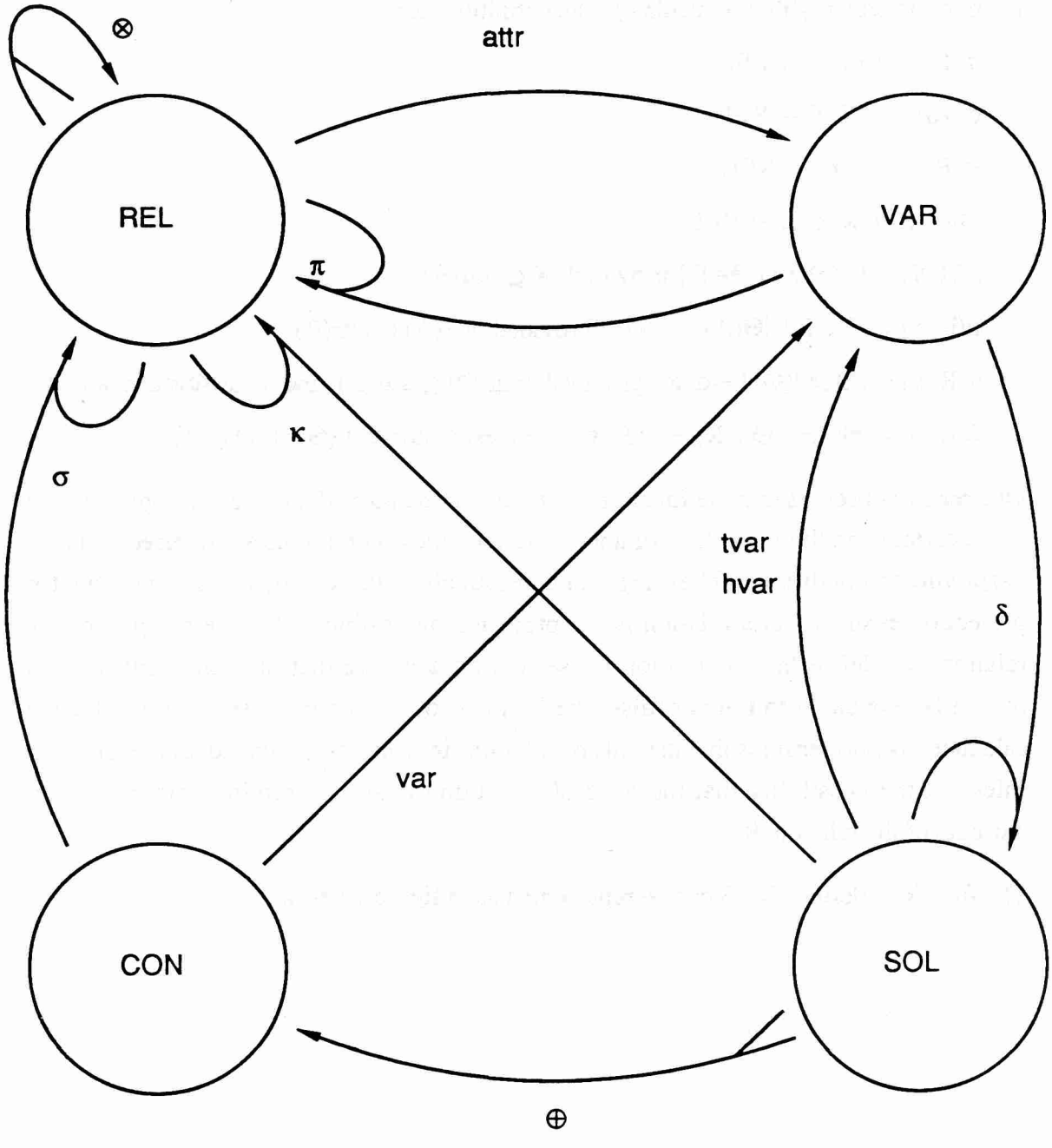

Fig. 1

Functions not listed in the diagram:

ρ: SOL × VAR → CON

_(_): SOL × SOL → SOL

_(_): SOL × CON → CON

4. Normal forms

In this section, we define normal forms and show that every expression of sort REL is equivalent to an expression in normal form.

4.1. Definition. A *normal form* is an expression of the form

$\pi(\kappa(\sigma(R_1 \otimes \ldots \otimes R_n, A), \Phi), X)$,

where $R_1,\ldots,R_n$ are atomic expressions. NF is the collection of normal forms.

4.2. Proposition. The normal form $\pi(\kappa(\sigma(R_1 \otimes \ldots \otimes R_n, A), \Phi), X)$ is defined iff:

1. $var(A) \subseteq attr(R_1 \otimes \ldots \otimes R_n)$
2. $tvar(\Phi) \subseteq attr(R_1 \otimes \ldots \otimes R_n)$
3. $X \subseteq attr(R_1 \otimes \ldots \otimes R_n) \cup hvar(\Phi)$
4. $attr(R_1 \otimes \ldots \otimes R_n) \cap hvar(\Phi) = \emptyset$

Proof. Follows directly from the definedness conditions.

4.3. Proposition. Consequences of the definedness of $\pi(\kappa(\sigma(R_1 \otimes \ldots \otimes R_n, A), \Phi), X)$ are:

1. $tvar(\Phi) \cap hvar(\Phi) = \emptyset$
2. $var(A) \cap hvar(\Phi) = \emptyset$
3. $hvar(\Phi) = (var(A) \cup var(\Phi)) - attr(R_1 \otimes \ldots \otimes R_n)$

Proof. Follows directly from the definedness conditions and 4.2.

4.4. Theorem. Every defined expression in **PCSJL** of sort REL can be transformed into an equivalent defined normal form.

Proof. Let NF be the collection of all expressions equivalent to a normal form (in other words, NF is the closure of NF under equivalence). It suffices to show the following statements:

(1) all atomic expressions of sort REL are in NF;

(2) NF is closed under π, σ, κ and $\otimes$.

This is done as follows.

(1) An atomic expression R can be rewritten in normal form by

$R = \pi(\kappa(\sigma(R, \mathbf{true}), \emptyset), attr(R))$.

(2) This splits in four cases: $(2.\pi)$, $(2.\sigma)$, $(2.\kappa)$ and $(2.\otimes)$. In the sequel, we use the following abbreviations:

$R := R_1 \otimes \ldots \otimes R_n$, $S := S_1 \otimes \ldots \otimes S_n$.

(2.π) This is easy: we have to show

$\pi(\pi(\kappa(\sigma(R,A),\Phi),X),Y) \in \underline{NF}$

and this follows from

$\pi(\pi(\kappa(\sigma(R,A),\Phi),X),Y) = \pi(\kappa(\sigma(R,A),\Phi),Y)$,

for it follows from the definedness conditions that $Y \subseteq X$.

(2.σ) This case is slightly more complex. We must show

$\sigma(\pi(\kappa(\sigma(R,A),\Phi),X),B) \in \underline{NF}$.

To obtain a defined normal form for this expression, we have to use substitution. The critical point in the construction is that variables from hvar(Φ) may be present in the constraint B. If B were added straightaway to the solution part of the new normal form, an undefined normal form would be constructed. Therefore the variables hvar(Φ) are substituted by the substitution in $\Phi(B)$, resulting in a defined normal form:

$\sigma(\pi(\kappa(\sigma(R,A),\Phi),X),B) = \pi(\kappa(\sigma(R, A \wedge \Phi(B)),\Phi),X)$.

(2.κ) Here we want

$\kappa(\pi(\kappa(\sigma(R,A),\Phi),X),\Psi) \in \underline{NF}$.

We begin with assuming that the following condition holds:

(*) $\text{hvar}(\Psi) \cap (\text{attr}(R) \cup \text{hvar}(\Phi)) - X = \emptyset$

For this case we can show that under assumption of (*) the next condition is true:

(**) $\text{hvar}(\Phi) \cap \text{hvar}(\Psi) = \emptyset$, (i.e. $\Phi \cup \Psi$ is a solution set)

For suppose $x \in \text{hvar}(\Phi) \cap \text{hvar}(\Psi)$. Then either $x \in X$ or $x \notin X$. If $x \in X$, the definedness conditions are violated since x is both determined by a head of a solution in Ψ and the relation $\pi(\kappa(\sigma(R,A),\Phi),X)$. On the other hand, if $x \notin X$ then (*) does not hold, contrary to our assumption. So it must be that the assumption $x \in \text{hvar}(\Phi) \cap \text{hvar}(\Psi)$ was false and hence (**) holds.

A second reason for adopting (*) is to ensure that the variables projected away by the projection on X, should be different from hvar(Ψ), the new variables introduced by Ψ. But this is already the case since we have assumed (*) to hold.

The resulting normal form is:

$\kappa(\pi(\kappa(\sigma(R,A),\Phi),X),\Psi) =$
$\pi(\kappa(\sigma(R,A\ ),\ \Phi \cup \Phi(\Psi)),\ X \cup hvar(\Psi))$

If (*) does not hold, then in R, A and Φ the variables in $(attr(R) \cup hvar(\Phi)) - X$ have to be renamed in order to make them different from those in $hvar(\Psi) - X$. After the renaming (*) holds and the rest of the argument runs analogously.

(2.$\otimes$) This last case is the most complex. What we want is

$\pi(\kappa(\sigma(R,A),\Phi),X) \otimes \pi(\kappa(\sigma(S,B),\Psi),Y) \in \underline{NF}$.

The condition to prevent an undesired clash of variables now reads :

(***) $(attr(R) \cup hvar(\Phi)) \cap (attr(S) \cup hvar(\Psi)) - (X \cap Y) = \emptyset$

We start with assuming that (***) holds. Then $\Phi \cup \Psi$ is, in general, not a solution set. The merge function $\oplus$ in the construction below handles this case. Also another case needs to be checked. Suppose there is a solution $x=t \in \Phi$ with $x \in attr(S)$. If this solution were put in the calculate part of the new normal form, then the resulting expression would be undefined. The problem can be handled by recognizing that x=t now satisfies the definedness condition of the select operator, viz. $var(x=t) \subseteq attr(R) \cup attr(S)$. So the restriction operator below inserts the solution x=t in the select condition of the new normal form and the delete operator deletes it from the calculate part. The symmetric case that there is a solution $x=t \in \Psi$ so that $x \in attr(R)$, is handled in the same way. The resulting normal form, which is defined, now reads:

$\pi(\kappa(\sigma(R,A),\Phi),X) \otimes \pi(\kappa(\sigma(S,B),\Psi),Y) =$
$\pi(\kappa(\sigma(R \otimes S,\ A \wedge B \wedge \Phi \oplus \Psi \wedge\ \rho(\Phi,attr(S)) \wedge \rho(\delta(\Psi,hvar(\Phi)),attr(R))\ ),$
$\qquad \delta(\Phi,attr(S)) \cup \delta(\delta(\Psi,hvar(\Phi)),attr(R))\ ),$
$\qquad X \cup Y\ )$

The above construction can be explained in the following way. The solution sets Φ and Ψ are transformed into the constraints $\Phi \oplus \Psi$ and the solution sets $\delta(\Psi,hvar(\Phi))$ and Φ. $\Phi \oplus \Psi$ is put directly into the select condition; the remaining pair of solution sets $\delta(\Psi,hvar(\Phi))$ and Φ needs to be processed further. On both solution sets the restriction ρ is applied to see whether more solutions can be turned into select conditions. The applied restrictions are compensated by the delete operators δ in the calculate part.

It should be noted that in the above construction, the expression $\rho(\delta(\Psi,hvar(\Phi)),attr(R))$ can be can be replaced by the more simple expression $\rho(\Psi,attr(R))$. However this could lead to duplicate use of solutions from Ψ in the condition of the resulting normal form and we want to avoid this duplication.

If (***) does not hold then we are going to rename variables. The condition (***) is

equivalent with the conjunction of the following two conditions:

(1) $((attr(R) \cup hvar(\Phi)) - X) \cap (attr(S) \cup hvar(\Psi)) = \emptyset$

(2) $(attr(R) \cup hvar(\Phi)) \cap ((attr(S) \cup hvar(\Psi)) - Y) = \emptyset$

First we rename in R, A and Φ the variables in $(attr(R) \cup hvar(\Phi)) - X$ in order to make them different from the variables $attr(S) \cup hvar(\Psi)$. After renaming (1) holds. In a similar way the offending variables for condition (2) are renamed in S, B and Ψ. After these renamings both (1) and (2) hold, so also (***) holds. The rest of the argument runs analogously.

This ends the proof of Theorem 4.4.

A normalisation construction for PSJ expressions can be obtained from the above normalization construction for PCSJ expressions by taking both $\Phi=\emptyset$ and $\Psi=\emptyset$. The calculate rule is dropped from the normalization construction.

5. Implementation

The normalisation procedure of the previous section has been implemented by the first author (in PROLOG). The sample problems listed below were run on this prototype. Output was adapted to the notation of this paper. The listed problems are examples of use of the join rule.

% conditions and constraints are combined directly:

$\pi(\kappa(\sigma(r(a,b),\{a>b\}),\{x=a-b\}),\{a,b,x\}) \otimes \pi(\kappa(\sigma(s(c,d),\{c>d\}),\{y=c-d\}),\{c,d,y\})$

$\rightarrow \quad \pi(\kappa(\sigma(r(a,b) \otimes s(c,d), a>b \wedge c>d),\{x=a-b,y=c-d\}),\{a,b,x,c,d,y\})$

% a solution is used as a condition in the new normal form:

$\pi(\kappa(\sigma(r(a,b),\{a>b\}),\{x=a-b\}),\{a,b,x\}) \otimes \pi(\kappa(\sigma(s(x,d),\{x>d\}),\{y=x-d\}),\{x,d,y\})$

$\rightarrow \quad \pi(\kappa(\sigma(r(a,b) \otimes s(x,d), a>b \wedge x>d \wedge x=a-b),\{y=x-d\}),\{a,b,x,d,y\})$

% a variable needs to be renamed:

$\pi(\kappa(\sigma(r(a,b),\{a>b\}),\{x=a-b\}),\{a,b,x\}) \otimes \pi(\kappa(\sigma(s(x,d),\{x>d\}),\{y=x-d\}),\{d,y\})$

$\rightarrow \quad \pi(\kappa(\sigma(r(a,b) \otimes s(u_1,d), a>b \wedge u_1>d),\{x=a-b,y=u_1-d\}),\{a,b,x,d,y\})$

% a variable occurs as a head in both solution sets:

$\pi(\kappa(\sigma(r(a,b),\{a>b\}),\{x=a-b\}),\{a,b,x\}) \otimes \pi(\kappa(\sigma(s(c,d),\{c>d\}),\{x=c-d\}),\{c,d,x\})$

$\rightarrow \quad \pi(\kappa(\sigma(r(a,b) \otimes s(c,d), a>b \wedge c>d \wedge a-b=c-d),\{x=a-b\}),\{a,b,x,c,d\})$

6. Conclusions

In this paper we have defined **PCSJL**, a language with expressions built up using the operations *projection*, *selection* , *join* and *calculate*. We have shown that a Normal Form Theorem for this language exists, by giving a construction to transform arbitrary relation expressions into normal form. Finally we mentioned a prototype system to demonstrate that the construction can practically be implemented.

There are several directions for further research in this area. First of all, the translation of **RL**-expressions into **PCSJL**-expressions is to be worked out, making use of a constraint solver: this is currently investigated by the authors. Another point is of a more theoretical nature, viz. to develop a more rigorous treatment of the reasoning in **PCSJL** by means of algebraic axiomatics: a challenging issue is formed by the renamings encountered in the proof of Theorem 4.4.

References

DATE87 Date, C.J., *A Guide to the SQL Standard*, Addison-Wesley Publishing Company 1987.

DATE88 Date, C.J. & White, C.J., *A Guide to DB2*, (Second Edition), Addison-Wesley Publishing Company 1988.

DEN88 van Denneheuvel, S. & van Emde Boas, P., *Towards implementing RL*, Preprint CT-88-11, Institute for Language, Logic and Information, University of Amsterdam, 1988

HANS87 Hansen, M.R., Hansen, B.S., Lucas, P. & van Emde Boas, P., *Integrating Relational Databases and Constraint Languages*, Rep. IBM Research, RJ 5594 (56904), 1987

HANS88 Hansen, M.R., *Algebraic Optimization of Recursive Database Queries*, Information Systems and Operations Research 26 (1988) 286-298

LAR85 Larson, P.A. & Yang, H.Z., *Computing Queries from Derived Relations* , Proc. of the 11th Intl. Conf. on VLDB, 259-269, (1985).

ULL82 Ullman, J.D., *Principles of Database Systems (Second Edition)*, Computer Science Press, 1982

VEMD86a van Emde Boas, P., *RL, a Language for Enhanced Rule Bases Database Processing*, Working Document, Rep IBM Research, RJ 4869 (51299), 1986.

VEMD86b van Emde Boas, P., *A semantical model for integration and modularization of rules*, Proceedings MFCS 12, Bratislava, August 1986, Springer Lecture Notes in Computer Science **233** (1986) 78-92

VEMD86c van Emde Boas, H. & van Emde Boas, P., *Storing and Evaluating Horn-Clause Rules in a Relational Database*, IBM J. Res. Develop. **30** (1986) 80-92

YAN87 Yang,H.Z, & Larson,P.A., *Query Transformations for PSJ-queries*, in Proc. of the 13th VLDB Conference, Brighton 1987, 245-254, (1987).

Logic Group Preprint Series

Department of Philosophy
University of Utrecht
Heidelberglaan 2
3584 CS Utrecht
The Netherlands

1 C.P.J. Koymans, J.L.M. Vrancken, *Extending Process Algebra with the empty process*, September 1985
2 J.A. Bergstra, *A process creation mechanism in Process Algebra*, September 1985
3 J.A. Bergstra, *Put and get, primitives for synchronous unreliable message passing*, October 1985
4 A. Visser, *Evaluation, provably deductive equivalence in Heyting's arithmetic of substitution instances of propositional formulas*, November 1985
5 G.R. Renardel de Lavalette, *Interpolation in a fragment of intuitionistic propositional logic*, January 1986
6 C.P.J. Koymans, J.C. Mulder, *A modular approach to protocol verification using Process Algebra*, April 1986
7 D. van Dalen, F.J. de Vries, *Intuitionistic free abelian groups*, April 1986
8 F. Voorbraak, *A simplification of the completeness proofs for Guaspari and Solovay's R*, May 1986
9 H.B.M. Jonkers, C.P.J. Koymans & G.R. Renardel de Lavalette, *A semantic framework for the COLD-family of languages*, May 1986
10 G.R. Renardel de Lavalette, *Strictheidsanalyse*, May 1986
11 A. Visser, *Kunnen wij elke machine verslaan? Beschouwingen rondom Lucas' argument*, July 1986
12 E.C.W. Krabbe, *Naess's dichotomy of tenability and relevance*, June 1986
13 H. van Ditmarsch, *Abstractie in wiskunde, expertsystemen en argumentatie*, Augustus 1986
14 A. Visser, *Peano's Smart Children, a provability logical study of systems with built-in consistency* , October 1986
15 G.R. Renardel de Lavalette, *Interpolation in natural fragments of intuitionistic propositional logic*, October 1986
16 J.A. Bergstra, *Module Algebra for relational specifications*, November 1986
17 F.P.J.M. Voorbraak, *Tensed Intuitionistic Logic*, January 1987
18 J.A. Bergstra, J. Tiuryn, *Process Algebra semantics for queues*, January 1987
19 F.J. de Vries, *A functional program for the fast Fourier transform*, March 1987
20 A. Visser, *A course in bimodal provability logic*, May 1987
21 F.P.J.M. Voorbraak, *The logic of actual obligation, an alternative approach to deontic logic*, May 1987
22 E.C.W. Krabbe, *Creative reasoning in formal discussion*, June 1987
23 F.J. de Vries, *A functional program for Gaussian elimination*, September 1987
24 G.R. Renardel de Lavalette, *Interpolation in fragments of intuitionistic propositional logic*, October 1987 (revised version of no. 15)
25 F.J. de Vries, *Applications of constructive logic to sheaf constructions in toposes*, October 1987
26 F.P.J.M. Voorbraak, *Redeneren met onzekerheid in expertsystemen*, November 1987
27 P.H. Rodenburg, D.J. Hoekzema, *Specification of the fast Fourier transform algorithm as a term rewriting system*, December 1987

28 D. van Dalen, *The war of the frogs and the mice, or the crisis of the Mathematische Annalen,* December 1987
29 A. Visser, *Preliminary Notes on Interpretability Logic,* January 1988
30 D.J. Hoekzema, P.H. Rodenburg, *Gauß elimination as a term rewriting system*, January 1988
31 C. Smoryński, *Hilbert's Programme,* January 1988
32 G.R. Renardel de Lavalette, *Modularisation, Parameterisation, Interpolation,* January 1988
33 G.R. Renardel de Lavalette, *Strictness analysis for POLYREC, a language with polymorphic and recursive types*, March 1988
34 A. Visser, *A Descending Hierarchy of Reflection Principles,* April 1988
35 F.P.J.M. Voorbraak, *A computationally efficient approximation of Dempster-Shafer theory*, April 1988
36 C. Smoryński, *Arithmetic Analogues of McAloon's Unique Rosser Sentences,* April 1988
37 P.H. Rodenburg, F.J. van der Linden, *Manufacturing a cartesian closed category with exactly two objects*, May 1988
38 P.H. Rodenburg, J.L.M.Vrancken, *Parallel object-oriented term rewriting : The Booleans*, July 1988
39 D. de Jongh, L. Hendriks, G.R. Renardel de Lavalette, *Computations in fragments of intuitionistic propositional logic*, July 1988
40 A. Visser, *Interpretability Logic,* September 1988
41 M. Doorman, *The existence property in the presence of function symbols,* October 1988
42 F. Voorbraak, *On the justification of Dempster's rule of combination*, December 1988
43 A. Visser, *An inside view of EXP, or: The closed fragment of the provability logic of* $I\Delta_0+\Omega_1$, February 1989
44 D.H.J. de Jongh & A. Visser, *Explicit Fixed Points in Interpretability Logic,* March 1989
45 S. van Denneheuvel & G.R. Renardel de Lavalette, *Normalisation of database expressions involving calculations*, March 1989