Relation-algebraic Computation of Fixed Points With Applications

Rudolf Berghammer [1]

Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität Kiel
Olshausenstraße 40, D-24098 Kiel

Abstract

We use relational algebra for describing sets and sets of sets and for computing all fixed points of certain functions on powersets. The method is illustrated by some examples from different problem domains, and results of our practical experiments with the RelView system are reported.

Key words: Relational algebra, Fixed point, Graph-theory, Cut completion, Permanent

1 Introduction

Computing fixed points is a central task in computer science and used, for example, in programming language semantics, algorithmics, dataflow analysis, model checking, and formal concept analysis. In most cases the task is to determine an extreme fixed point of a monotone or continuous function on a specific class of partially ordered sets. The basic fixed point theorems are the well-known Knaster-Tarski theorem [17] dealing with the existence of extreme fixed points in the case of complete lattices and its extension to complete partial orders [8,12]. Further generalizations of the Knaster-Tarski theorem can be found e.g., in [5] (set-valued functions) and [4] (monotone relations).

There are, however, situations where the classical fixed point theorems cannot be applied since, for instance, the function under consideration is non-monotone or one has to compute all fixed points of a function. We have iden-

Email address: rub@informatik.uni-kiel.de (Rudolf Berghammer).
[1] Fax: +49 431 880 7613

tified a lot of problems which can be reduced to the latter case and where, in addition, the function works on a powerset. In this paper we show how relational algebra [16,14] and a sophisticated implementation of it can be used to solve such – of course, usually computationally hard – problems. The method is based on a relation-algebraic description of sets and sets of sets and on modeling a function $F : 2^X \to 2^X$ on a powerset by a relational counterpart f. It can be applied if, in Boolean matrix terminology, the application of f to the membership-relation between X and 2^X yields a matrix the columns of which describe all possible results of F. Due to the use of the membership-relation the proposed procedure for computing all fixed points of F will require exponential time and space and, at a first glance, seems to be applicable only to small sets X. Nevertheless, many practical experiments with the relation-algebraic manipulation and prototyping system RelView [2,1] have shown that it often works sufficiently well, even in the case of sets X of medium size, if relational algebra is implemented using reduced ordered binary decision diagrams (abbreviated: ROBDDs), as described in [11,3,13].

The remainder of the paper is organized as follows: Some relation-algebraic preliminaries are recalled in Section 2. Section 3 shows how to describe sets and sets of sets using relational algebra and how to compute all fixed points of a function on a powerset by combining this description with a relation-algebraic modeling of the function. Some applications of the method are presented in Section 4. Their realization using RelView is explained in Section 5. This section also reports on the results of the numerous practical experiments with this system. Section 6 contains some concluding remarks.

2 Relation-algebraic Preliminaries

In the sequel, we introduce the basics of the algebra of set-theoretic relations and specify the classes of relations that are used in remainder of the paper. For more details, see [14] for example.

If X and Y are sets, then a subset R of the Cartesian product $X \times Y$ is called a relation with *domain* X and *range* Y. We denote the set (in this context also called *type*) of all relations with domain X and range Y by $[X \leftrightarrow Y]$ and write $R : X \leftrightarrow Y$ instead of $R \in [X \leftrightarrow Y]$. If X and Y are finite sets of size m and n, respectively, then we may consider a relation $R : X \leftrightarrow Y$ as a Boolean matrix with m rows and n columns. The Boolean matrix interpretation of relations is well suited for many purposes and also used as graphical representation within the RelView system. Therefore, in this paper we often use Boolean matrix terminology and notation. Especially, we speak about the rows and columns of R and write R_{xy} instead of the formula $(x, y) \in R$.

We assume the reader to be familiar with the basic operations on relations, viz. R^T (*transposition*), $\overline{R}$ (*complementation*), $R \cup S$ (*union*), $R \cap S$ (*intersection*), $R\,S$ (*composition*), $R \subseteq S$ (*inclusion*), and the special relations O (*empty relation*), L (*universal relation*), and I (*identity relation*).

Let R be a homogeneous relation, i.e., a relation for which domain and range coincide. Then R is *reflexive* if $\mathsf{I} \subseteq R$, *irreflexive* if $R \subseteq \overline{\mathsf{I}}$, *transitive* if $RR \subseteq R$, *antisymmetric* if $R \cap R^\mathsf{T} \subseteq \mathsf{I}$, and *symmetric* if $R = R^\mathsf{T}$. A *quasi-order* is a reflexive and transitive relation and a *partial order* is antisymmetric in addition. Now, assume R to be an arbitrary (also called heterogeneous) relation. Then R is *univalent* if $R^\mathsf{T}R \subseteq \mathsf{I}$ and *total* if $R\mathsf{L} = \mathsf{L}$. As usual, a univalent and total relation is called a *mapping*. A relation R is *injective* if R^T is univalent and *surjective* if R^T is total.

Given two relations $R : X \leftrightarrow Y$ and $S : X \leftrightarrow Z$, their *right residual* $R \setminus S : Y \leftrightarrow Z$ and *symmetric quotient* $\mathrm{syq}(R, S) : Y \leftrightarrow Z$ are defined by

$$R \setminus S := \overline{R^\mathsf{T}\overline{S}} \qquad \mathrm{syq}(R, S) := \overline{R^\mathsf{T}\overline{S}} \cap \overline{\overline{R}^\mathsf{T} S}.$$

In component-wise notation we have $(R \setminus S)_{yz}$ if and only if $R_{xy} \to S_{xz}$ for all $x \in X$ and $\mathrm{syq}(R, S)_{yz}$ if and only if $R_{xy} \leftrightarrow S_{xz}$ for all $x \in X$.

3 Computation of Fixed Points

Relational algebra offers some simple and elegant ways of describing sets and sets of sets. In this section, first, we present three possibilities. Then, we show how they can be used to compute all fixed points of functions on powersets.

3.1 Describing Sets and Sets of Sets With Relations

The first description of sets uses *vectors*, which are relations with $v = v\mathsf{L}$. For $v : X \leftrightarrow Y$ this condition means: Whatever set Z and universal relation $\mathsf{L} : Y \leftrightarrow Z$ we choose, an element $x \in X$ is either in relationship $(v\mathsf{L})_{xz}$ to no element $z \in Z$ or to all elements $z \in Z$. As for a vector, therefore, the range is irrelevant, we consider in the following mostly vectors $v : X \leftrightarrow \mathbf{1}$ with a specific singleton set $\mathbf{1} = \{\bot\}$ as range and omit in such cases the second subscript, i.e., write v_x instead of $v_{x\bot}$. A vector $v : X \leftrightarrow \mathbf{1}$ can be considered as a Boolean matrix with exactly one column, i.e., as a Boolean column vector, and *describes* (or: is a description of) the subset $\{x \in X \mid v_x\}$ of X.

A vector v is said to be a *point* if it is injective and surjective. These properties

mean that it describes a singleton subset of its domain X or an element from X if we identify a singleton set with the only element it contains. In the Boolean matrix model, hence, a point $v : X \leftrightarrow \mathbf{1}$ is a Boolean column vector in which exactly one entry is *true*.

We will use injective mappings as our second way of describing sets. Given an injective mapping $\imath : Y \leftrightarrow X$, we may consider Y as a subset of X by identifying it with its image under $\imath$. If Y actually is a subset of X and $\imath$ is the identity mapping from Y to X, then the vector $\imath^{\mathsf{T}}\mathsf{L} : X \leftrightarrow \mathbf{1}$ describes Y as a subset of X in the above sense. Clearly, also the transition in the other direction is possible, i.e., the generation of the injective mapping

$$\mathrm{inj}(v) : Y \leftrightarrow X \qquad\qquad \mathrm{inj}(v)_{yx} \;:\Longleftrightarrow\; y = x$$

from a given vector $v : X \leftrightarrow \mathbf{1}$ describing the subset Y of X. We call $\mathrm{inj}(v)$ the *injective mapping generated* by v.

As our third possibility to describe sets, we will use the relation-level equivalents of the set-theoretic symbol "$\in$", that is, the *membership-relation*

$$\mathsf{M} : X \leftrightarrow 2^X \qquad\qquad \mathsf{M}_{xY} \;:\Longleftrightarrow\; x \in Y\,.$$

A Boolean matrix representation of M requires exponential space. However, in [11] a ROBDD-implementation of M is developed the number of vertices of which is linear in the size of X.

Using Boolean matrix terminology, membership-relations lead to a column-wise description of sets of sets. More specifically, if the vector $v : 2^X \leftrightarrow \mathbf{1}$ describes a subset $\mathfrak{S}$ of 2^X in the above sense, i.e., v_Y if and only if $Y \in \mathfrak{S}$ for all $Y \in 2^X$, then for all $x \in X$ and $Y \in \mathfrak{S}$ we get the equivalence of $x \in Y$ and $(\mathsf{M}\,\mathrm{inj}(v)^{\mathsf{T}})_{xY}$. This means that the elements of $\mathfrak{S}$ are described precisely by the columns of the relational composition $\mathsf{M}\,\mathrm{inj}(v)^{\mathsf{T}} : X \leftrightarrow \mathfrak{S}$.

Let $R : X \leftrightarrow X$ be a quasi-order and $v : X \leftrightarrow \mathbf{1}$ describe a subset Y of X. Combining the latter fact with some well-known correspondences between relation-algebraic and and logical constructions, we are able to show

$$\begin{aligned} x \in Y \wedge \forall\, y : y \in Y \rightarrow R_{yx} &\iff v_x \wedge \forall\, y : v_y \rightarrow R_{yx} \\ &\iff v_x \wedge \neg\exists\, y : v_y \wedge \overline{R}^{\mathsf{T}}_{xy} \\ &\iff v_x \wedge (\overline{\overline{R^{\mathsf{T}}}\,v})_x \\ &\iff (v \cap \overline{\overline{R^{\mathsf{T}}}\,v})_x \end{aligned}$$

for all $x \in X$. This calculation implies that the set of *greatest elements* of

Y with respect to R (note that in the case of quasi-orders greatest elements must not be unique) is described by the vector

$$GreEl(R,v) := v \cap \overline{\overline{R}^{\mathsf{T}} v} : X \leftrightarrow \mathbf{1}\,.$$

In Section 4 we will apply this construction to compute largest sets. This means that X is the powerset 2^Y of some set Y, the first argument of the relational function *GreEl* is the *size-comparison-relation*

$$\mathsf{C} : 2^Y \leftrightarrow 2^Y \qquad\qquad \mathsf{C}_{AB} \;:\Longleftrightarrow\; |A| \leq |B|\,,$$

and the second argument of *GreEl* is a vector describing that subset of 2^Y the largest elements of which we are interested in. ROBDDs allow a very compact implementation of size-comparison-relations. The number of vertices of the ROBDD representing C is quadratic in the size of Y; see [13].

3.2 Computation of Fixed Points of Functions on Powersets

In this and the following subsection we fix a non-empty set X and a universe $\mathfrak{U}$ containing at least the sets $\mathbf{1}$ and 2^X. By $\mathcal{R}[X]$ we denote the set of all relations with domain X and a range from $\mathfrak{U}$. The set X and all sets from $\mathfrak{U}$ are assumed to be finite. The only reason for this restriction is that we want to describe algorithms by relation-algebraic expressions that are executable using a system like RelView. It is never applied in a formal calculation.

Assume a function $F : 2^X \to 2^X$. Furthermore, let $f : \mathcal{R}[X] \to \mathcal{R}[X]$ be a relational function such for all $R \in \mathcal{R}[X]$ the types of R and $f(R)$ coincide. We say that F is *modeled* by f if $f(v)$ describes $F(Y)$ for all $v : X \leftrightarrow \mathbf{1}$ and $Y \in 2^X$ such that v describes Y. If, in addition, the equivalence

$$x \in F(Y) \iff f(\mathsf{M})_{xY} \tag{1}$$

holds for the membership-relation $\mathsf{M} : X \leftrightarrow 2^X$ and all $x \in X$ and $Y \in 2^X$, then the function F is said to be *modeled column-wise* by f. This means that the set of sets $\{F(Y) \mid Y \in 2^X\}$ is described column-wise by the relation $f(\mathsf{M}) : X \leftrightarrow 2^X$ as explained in Section 3.1.

Our aim is to develop a vector of type $[2^X \leftrightarrow \mathbf{1}]$, which describes the set $\mathfrak{F}_F := \{Y \in 2^X \mid Y = F(Y)\}$ of fixed points of the function F. In doing so, we assume that F is modeled column-wise by the relational function f and apply, as in the derivation of the relational function *GreEl* in Section 3.1, predicate logic and correspondences between relation-algebraic and logical

constructions. For all $Y \in 2^X$ we calculate as follows, where the type of the universal vector L introduced in the fifth step is $[X \leftrightarrow \mathbf{1}]$:

$$
\begin{aligned}
& Y = F(Y) \\
\iff & \forall\, x : x \in Y \leftrightarrow x \in F(Y) \\
\iff & \forall\, x : \mathsf{M}_{xY} \leftrightarrow f(\mathsf{M})_{xY} \qquad (1) \\
\iff & \forall\, x : (\mathsf{M}_{xY} \to f(\mathsf{M})_{xY}) \wedge (f(\mathsf{M})_{xY} \to \mathsf{M}_{xY}) \\
\iff & \forall\, x : (\neg\mathsf{M}_{xY} \vee f(\mathsf{M})_{xY}) \wedge (\neg f(\mathsf{M})_{xY} \vee \mathsf{M}_{xY}) \\
\iff & \neg\exists\, x : \neg((\neg\mathsf{M}_{xY} \vee f(\mathsf{M})_{xY}) \wedge (\neg f(\mathsf{M})_{xY} \vee \mathsf{M}_{xY})) \wedge \mathsf{L}_x \\
\iff & \neg\exists\, x : \neg((\neg\mathsf{M}^{\mathsf{T}}_{Yx} \vee f(\mathsf{M})^{\mathsf{T}}_{Yx}) \wedge (\neg f(\mathsf{M})^{\mathsf{T}}_{Yx} \vee \mathsf{M}^{\mathsf{T}}_{Yx})) \wedge \mathsf{L}_x \\
\iff & \neg\exists\, x : \overline{(\overline{\mathsf{M}^{\mathsf{T}}} \cup f(\mathsf{M})^{\mathsf{T}}) \cap (\overline{f(\mathsf{M})^{\mathsf{T}}} \cup \mathsf{M}^{\mathsf{T}})}_{Yx} \wedge \mathsf{L}_x \\
\iff & \overline{\overline{(\overline{\mathsf{M}^{\mathsf{T}}} \cup f(\mathsf{M})^{\mathsf{T}}) \cap (\overline{f(\mathsf{M})^{\mathsf{T}}} \cup \mathsf{M}^{\mathsf{T}})\mathsf{L}}}_{Y} .
\end{aligned}
$$

Now, we remove the subscript Y from the last expression of this calculation following the vector description of sets (cf. Section 3.1) and apply after that de Morgan's law and some well-known rules to exchange transposition of relations with relational composition and the Boolean operations on relations. This yields the vector description of the set $\mathfrak{F}_F$ as $\mathit{FixPts}(f(\mathsf{M})) : 2^X \leftrightarrow \mathbf{1}$, where the relational function $\mathit{FixPts} : [X \leftrightarrow 2^X] \to [2^X \leftrightarrow \mathbf{1}]$ is defined as follows:

$$\mathit{FixPts}(Q) = \overline{\mathsf{L}^{\mathsf{T}}((\mathsf{M} \cap \overline{Q}) \cup (Q \cap \overline{\mathsf{M}}))}^{\mathsf{T}} .$$

Compared with the last expression of the calculation above, in the expression defining the relational function *FixPts* only the small universal relation $\mathsf{L} : X \leftrightarrow \mathbf{1}$ and a relation of type $[\mathbf{1} \leftrightarrow 2^X]$ (a "row vector") are transposed instead of relations of type $[X \leftrightarrow 2^X]$. This is very advantageous in case of a ROBDD-implementation of relational algebra since here transposition of a relation with domain or range $\mathbf{1}$ does not affect the ROBDD. Only domain and range (which in RelView are represented by numbers) are exchanged. See [13] for details.

From the result of the call $\mathit{FixPts}(f(\mathsf{M}))$ a description of the set $\mathfrak{F}_F$ of all fixed points as the columns of a relation of type $[X \leftrightarrow \mathfrak{F}_F]$ easily can be obtained using the procedure described in Section 3.1.

3.3 Calculating Relational Functions That Model Column-wise

Given a function $F : 2^X \to 2^X$, in general the development of the modeling relational function f follows the strategy we have already applied in Sections 3.1

and 3.2. One assumes the set $Y \in 2^X$ to be described by the vector $v : X \leftrightarrow \mathbf{1}$. Starting with the relationship $x \in F(Y)$ and using correspondences between relation-algebraic and logical constructions in combination with laws of predicate logic and relational algebra, then one calculates a relation-algebraic expression $\mathcal{E}(v)$ of type $[X \leftrightarrow \mathbf{1}]$ such that $x \in F(Y)$ if and only if $\mathcal{E}(v)_x$. Because of this equivalence, F is modeled by

$$f : \mathcal{R}[X] \to \mathcal{R}[X] \qquad\qquad f(v) = \mathcal{E}(v)\,. \tag{2}$$

If, in addition, the body $\mathcal{E}(v)$ of the relational function f of (2) is constructed following the grammar [2]

$$\mathcal{E} ::= v \mid \overline{\mathcal{E}} \mid \mathcal{E} \cup \mathcal{E} \mid \mathcal{E} \cap \mathcal{E} \mid R\,\mathcal{E},$$

where R is a relation-algebraic expression free of v, then the function F is even modeled column-wise by f (and, hence, the relational function *FixPts* can be applied for computing F's fixed points). In the remainder of this section, we will prove this result. In doing so, we denote the replacement of v in $\mathcal{E}(v)$ by S as $\mathcal{E}(S)$ and use the same notation also for the replacement in sub-expressions of $\mathcal{E}(v)$.

Assume $x \in X$ and $Y \in 2^X$. Furthermore, let the set Y be described by the vector $w : X \leftrightarrow \mathbf{1}$. (Using set-theoretic notation, w corresponds to the direct product $Y \times \mathbf{1}$.) Since F is modeled by f, the vector $f(w) : X \leftrightarrow \mathbf{1}$ describes the set $F(Y) \in 2^X$. This implies that $f(w)_x$ and $x \in F(Y)$ are equivalent. Combining this fact with definition (2), property (1) follows from

$$\mathcal{E}(w)_x \iff \mathcal{E}(\mathsf{M})_{xY}\,. \tag{3}$$

To prove (3) for all $x \in X$, $Y \in 2^X$, and vector descriptions $w : X \leftrightarrow \mathbf{1}$ of Y, we use induction on the structure of $\mathcal{E}(v)$ given by the above grammar. For the induction base we assume that $\mathcal{E}(v)$ is the parameter v. Then we obtain

$$w_x \iff x \in Y \iff \mathsf{M}_{xY}$$

due to the description of Y by w and the specification of M. This shows the desired result since the expression $\mathcal{E}(w)$ equals w and the expression $\mathcal{E}(\mathsf{M})$ equals M. As first case of the induction step we consider complementation, i.e., assume $\mathcal{E}(v)$ to be of the form $\overline{\mathcal{E}'(v)}$. Here we have

$$\overline{\mathcal{E}'(w)}_x \iff \neg(\mathcal{E}'(w)_x) \iff \neg(\mathcal{E}'(\mathsf{M})_{xY}) \iff \overline{\mathcal{E}'(\mathsf{M})}_{xY}\,,$$

[2] In words this means that the body of f is constructed only with the help of the parameter v, the Boolean operations on relations, and the specific composition $S \mapsto RS$, where v does not occur in R.

where the induction hypothesis is applied in the second step. In the same way one deals with union and intersection. It remains the case that $\mathcal{E}(v)$ is a composition $R\,\mathcal{E}'(v)$ with R being free from v. Here (3) is shown by

$$\begin{aligned}(R\,\mathcal{E}'(w))_x &\iff \exists\, y : R_{xy} \wedge \mathcal{E}'(w)_y \\ &\iff \exists\, y : R_{xy} \wedge \mathcal{E}'(\mathsf{M})_{yY} \\ &\iff (R\,\mathcal{E}'(\mathsf{M}))_{xY}\,,\end{aligned}$$

where the induction hypothesis again is applied in the second step.

4 Applications

This section is devoted to applications. We treat problems from different domains and demonstrate how to solve them using the method of Section 3.

4.1 Cut Completions

Assume (X, R) to be a partially ordered set, i.e., $R : X \leftrightarrow X$ to be a partial order. For a set $Y \in 2^X$ let $Y^{\downarrow} := \{x \in X \mid \forall\, y \in Y : R_{xy}\}$ denote the set of its lower bounds and $Y^{\uparrow} := \{x \in X \mid \forall\, y \in Y : R_{yx}\}$ denote the set of its upper bounds with respect to R. A fixed point of the function

$$F : 2^X \to 2^X \qquad\qquad F(Y) = Y^{\uparrow\downarrow} \tag{4}$$

is called a *Dedekind cut* of (X, R). Obviously, for each element $x \in X$ the set $[x] := \{y \in X \mid R_{yx}\}$ is a Dedekind cut of (X, R), called the *principal cut* generated by x. Now, let $\mathfrak{C}$ denote the set of all Dedekind cuts of (X, R) and $\mathfrak{P}$ be the subset of all principal cuts. Then $(\mathfrak{C}, \subseteq)$ is a complete lattice. It is called the *cut completion* of (X, R), since it contains $(\mathfrak{P}, \subseteq)$ as sub-order and the latter is order-isomorphic to (X, R) via the injective function $\sigma : X \to \mathfrak{C}$, mapping x to the principal cut generated by x.

For developing a relational function f, which models the function F of (4), we assume a set $Y \in 2^X$ and a vector description $v : X \leftrightarrow \mathbf{1}$ of Y. Then, we

have for all $x \in X$:

$$\begin{aligned}
x \in F(Y) &\iff x \in Y^{\uparrow\downarrow} && (4)\\
&\iff \forall\, y : y \in Y^{\uparrow} \to R_{xy} && \text{def. lower bound}\\
&\iff \forall\, y : (\forall\, z : z \in Y \to R_{zy}) \to R_{xy} && \text{def. upper bound}\\
&\iff \forall\, y : (\forall\, z : v_z \to R_{zy}) \to R_{xy} && v \text{ describes } Y\\
&\iff \forall\, y : (\neg\exists\, z : v_z \wedge \overline{R}_{zy}) \to R_{xy}\\
&\iff \forall\, y : (\overline{\overline{R}^{\mathsf{T}} v})_y \to R_{xy}\\
&\iff \neg\exists\, y : (\overline{\overline{R}^{\mathsf{T}} v})_y \wedge \overline{R}_{xy}\\
&\iff (\overline{R\,\overline{\overline{R}^{\mathsf{T}} v}})_x\,.
\end{aligned}$$

This calculation suggests to define f by $f(v) = \overline{R\,\overline{\overline{R}^{\mathsf{T}} v}}$. By virtue of the criterion shown in Section 3.3, the function F is even modeled column-wise by f. As a consequence, the vector

$$\mathit{Cut}(R) := \mathit{FixPts}(\overline{R\,\overline{\overline{R}^{\mathsf{T}}\mathsf{M}}}) : 2^X \leftrightarrow \mathbf{1}$$

describes the set $\mathfrak{C}$ and

$$\mathit{CutList}(R) := \mathsf{M}\,\mathrm{inj}(\mathit{Cut}(R))^{\mathsf{T}} : X \leftrightarrow \mathfrak{C}$$

is a column-wise description of $\mathfrak{C}$. Note that in both cases M stands for the membership-relation of type $[X \leftrightarrow 2^X]$.

Dedekind cuts are ordered by inclusion. Using the component-wise specification of right residuals in Section 2 in combination with the equivalence of $x \in Y$ and $\mathit{CutList}(R)_{xY}$, we are able to show

$$Y \subseteq Z \iff \forall\, x : x \in Y \to x \in Z \iff (\mathit{CutList}(R) \setminus \mathit{CutList}(R))_{YZ}$$

for all Dedekind cuts Y, Z. An immediate consequence of this property is the relation-algebraic version

$$\mathit{CutOrd}(R) := \mathit{CutList}(R) \setminus \mathit{CutList}(R) : \mathfrak{C} \leftrightarrow \mathfrak{C}$$

of the inclusion order on $\mathfrak{C}$. Also the embedding of (X, R) into its cut completion $(\mathfrak{C}, \subseteq)$ can be formulated quite easily using relational algeba. Given

$x \in X$ and $Y \in \mathfrak{C}$, we obtain

$$[x] = Y \iff \forall\, y : R_{yx} \leftrightarrow y \in Y \iff \mathrm{syq}(R, \mathit{cutset}(R))_{xY}\,,$$

where the first step uses the definition of principal cuts and the second step applies the component-wise specification of symmetric quotients in Section 2 and the equivalence of $y \in Y$ and $\mathit{CutList}(R)_{yY}$. This leads to

$$\mathit{Embedding}(R) := \mathrm{syq}(R, \mathit{CutList}(R)) : X \leftrightarrow \mathfrak{C}$$

as relation-algebraic version of the embedding function $\sigma : X \to \mathfrak{C}$.

4.2 Perfect Matchings and Permanents of 0/1-Matrices

We now concentrate on an important problem concerning 0/1-matrices. Let $A := (a_{ij})_{1 \le i,j \le n}$ be a $n \times n$ 0/1-matrix. The *permanent* of A is defined as

$$\mathrm{per}(A) := \sum_{\pi} \prod_{i=1}^{n} a_{i,\pi(i)}\,,$$

where the sum is over all permutations π on the set $\{1, \ldots, n\}$. This number has an important graph-theoretic interpretation. If we consider the bipartite undirected graph $g_A = (V, E)$ with $\{1, \ldots, 2n\}$ as set V of vertices and the set E of edges defined by $\{x, y\} \in E$ if and only if $1 \le x \le n < y \le 2n$ and $a_{x,y-n} = 1$, then it is not hard to verify that $\mathrm{per}(A)$ equals the number of perfect matchings of g_A. This property is the base of Sinclair's well-known approximation algorithm [10,15], which can be seen as a starting point of approximative counting, and also of our (exact) computation of $\mathrm{per}(A)$.

A *matching* of an undirected graph is a set of edges such that no two distinct edges have a vertex in common. It is called *perfect* if, in addition, every vertex of the graph is contained in precisely one of its edges. In the case of g_A this means that the set of perfect matchings coincides with the set of maximum [3] matchings if and only if n is the cardinality of the maximum matchings. If, however, each maximum matching M of g_A has cardinality $|M| < n$, then g_A has no perfect matching.

For computing all matchings of g_A by means of the relational function *FixPts*, we assume that g_A is represented by its *incidence relation* $R : V \leftrightarrow E$, which

[3] As usual, "maximum" means a largest set with respect to cardinality. If a set is a largest one with respect to inclusion, it is called "maximal".

describes the set E column-wise. That is, for all $x \in V$ and $a \in E$ we have R_{xa} if and only if $x \in a$. For reasons of simplification, furthermore, we consider $K := R^\mathsf{T} R \cap \overline{\mathsf{I}} : E \leftrightarrow E$, in [14] called *edge-adjacency relation*.

Now, assume $Y \in 2^E$ to be a set of edges. Then Y is a matching of g_A if and only if $\overline{K}_{ab}$ for all $a, b \in Y$. This, in turn, is equivalent to Y being a subset of $\{a \in E \mid \forall\, b \in Y : \overline{K}_{ab}\}$. The desired fixed point representation is an immediate consequence of this property: the set of all matchings of g_A is given precisely by the set of all fixed points of the function

$$F : 2^E \to 2^E \qquad F(Y) = Y \cap \{a \in E \mid \forall\, b \in Y : \overline{K}_{ab}\}\,. \tag{5}$$

It is not hard to calculate from definition (5) a relational function f, which models F. If the vector $v : E \leftrightarrow \mathbf{1}$ describes the set $Y \in 2^E$, then we have

$$\begin{aligned}
a \in F(Y) &\iff a \in Y \wedge \forall\, b \in Y : \overline{K}_{ab} && (5)\\
&\iff v_a \wedge \forall\, b : v_b \to \overline{K}_{ab} && v \text{ describes } Y\\
&\iff v_a \wedge \neg \exists\, b : v_b \wedge K_{ab}\\
&\iff v_a \wedge (\overline{Kv})_a\\
&\iff (v \cap \overline{Kv})_a
\end{aligned}$$

for all $a \in E$, yielding the definition $f(v) = v \cap \overline{Kv}$. This function even models F column-wise due to the criterion of Section 3.3. Hence, using the membership-relation $\mathsf{M} : E \leftrightarrow 2^E$ and the definition of the relation K we get $\mathit{FixPts}(\mathsf{M} \cap \overline{(R^\mathsf{T} R \cap \overline{\mathsf{I}})\mathsf{M}}) : 2^E \leftrightarrow \mathbf{1}$ as vector description of the set of all matchings of g_A. From this, the following application of the relational function *GreEl*, where the first argument is the size-comparison-relation on 2^E, yields a vector that describes the set of all maximum matchings of g_A:

$$\mathit{MaxMatch}(R) := \mathit{GreEl}(\mathsf{C}, \mathit{FixPts}(\mathsf{M} \cap \overline{(R^\mathsf{T} R \cap \overline{\mathsf{I}})\mathsf{M}})) : 2^E \leftrightarrow \mathbf{1}$$

As already mentioned, $\mathrm{per}(A)$ equals the cardinality of this vector (in Boolean matrix terminology: the number of its *true*-entries) if each element (i.e., maximum matching) from the subset $\mathfrak{M}$ of 2^E it describes has cardinality n. Otherwise, $\mathrm{per}(A) = 0$. RelView is able to compute cardinalities of relations. Using the system, also the test that all matchings of the set $\mathfrak{M}$ consist of n edges easily can be performed as follows: First, one takes some point $p : 2^E \leftrightarrow \mathbf{1}$ contained in the vector $\mathit{MaxMatch}(R)$. It describes a single maximum matching M of the graph g_A. After this selection one computes $\mathsf{M}p : E \leftrightarrow \mathbf{1}$, with $\mathsf{M} : E \leftrightarrow 2^E$ as membership-relation. Since for all edges $a \in E$ we have the

equivalence

$$(\mathsf{M}p)_a \iff \exists\, Z : \mathsf{M}_{aZ} \wedge p_Z \iff \exists\, Z : a \in Z \wedge Z = M \iff a \in M\,,$$

the vector $\mathsf{M}p$ describes M, too, and a comparison of its cardinality with n yields the desired result.

4.3 Independence Numbers and Some Further Graph-Parameters

In Section 4.2 we have shown how to compute matchings and the size of maximum matchings. This so-called *matching number* $\nu(g)$ of an undirected graph g is an important graph-parameter. Further important graph-parameters are, for example, the cardinality of all maximum independent sets of vertices (*independence number* $\alpha(g)$), of all maximum cliques (*clique number* $\omega(g)$), of all minimum vertex covers (*vertex cover number* $\tau(g)$), and of all minimum colourings (*chromatic number* $\chi(g)$). In the following, we show how to compute these numbers using our approach. In doing so, we assume the undirected and loop-free graph $g = (V, E)$ to be represented by its symmetric and irreflexive *adjacency relation* $R : V \leftrightarrow V$. That is, we suppose for all $x, y \in V$ that R_{xy} if and only if $\{x, y\} \in E$.

First, we investigate the computation of the independence number $\alpha(g)$. A set $Y \in 2^V$ is *independent* if no pair of distinct vertices $x, y \in Y$ is connected via an edge. This property holds precisely if the set $\{x \in V \mid \exists\, y \in Y : R_{xy}\}$ is contained in the complement of Y. Because of this inclusion it is obvious that a set is independent if and only if it is a fixed point of

$$F : 2^V \to 2^V \qquad\qquad F(Y) = Y \cap \{x \in V \mid \forall\, y \in Y : \overline{R}_{xy}\}\,. \tag{6}$$

This function is very similar to the function of (5); only the set E and the relation K are replaced by V and R, respectively. As a consequence, F is modeled column-wise by the relational function $f(v) = v \cap \overline{Rv}$. This yields

$$\mathit{MaxInd}(R) := \mathit{GreEl}(\mathsf{C}, \mathit{FixPts}(\mathsf{M} \cap \overline{R\mathsf{M}})) : 2^V \leftrightarrow \mathbf{1}$$

as vector description of the set of all maximum independent sets of g, where the membership-relation M has type $[V \leftrightarrow 2^V]$ and the size-comparison-relation C has type $[2^V \leftrightarrow 2^V]$. As shown at the end of Section 4.2, by means of a point $p \subseteq \mathit{MaxInd}(R)$ the independence number $\alpha(g)$ is obtained as cardinality of $\mathsf{M}p : V \leftrightarrow \mathbf{1}$, i.e., as number of *true*-entries of this vector.

The similarity of the functions of (5) and (6) is caused by the fact that M is a matching of g if and only if it is independent in g's *line-graph* (where

the vertices are the edges of g and two edges of g form an edge in the line graph if and only if they are distinct and have a vertex of g in common). Also the computation of all cliques of g can be reduced to the computation of all indepedent sets, since $Y \in 2^V$ is a clique of g if and only if it is independent in its *complement* $\overline{g} = (V, \overline{E})$, where $\overline{E} := \{\{x, y\} \mid x, y \in V, x \neq y, \{x, y\} \notin E\}$. The adjacency relation of $\overline{g}$ is $\overline{R \cup \mathsf{I}}$. Hence, the vector

$$\mathit{MaxClique}(R) := \mathit{GreEl}(\mathsf{C}, \mathit{FixPts}(\mathsf{M} \cap \overline{\overline{R \cup \mathsf{I}}\,\mathsf{M}})) : 2^V \leftrightarrow \mathbf{1}$$

describes the set of all maximum cliques of the graph g and the clique number $\omega(g)$ equals the cardinality of the vector $\mathsf{M}p : V \leftrightarrow \mathbf{1}$, where p is some point contained in $\mathit{MaxClique}(R)$.

The correspondence between independent sets and cliques is well-known in complexity theory and in textbooks often applied for reducing one problem to the other when proving NP-completeness. Also the vertex-cover-problem has the same complexity: it is known that a set $Y \in 2^V$ is a (minimum) vertex cover of the graph g if and only if its complement $V \setminus Y$ is a (maximum) independent set of g. Using this fact and the membership-relation $\mathsf{M} : V \leftrightarrow 2^V$, we can compute the vertex cover number $\tau(g)$ by selecting a point $p \subseteq \mathit{MaxInd}(R)$ and determining after that the cardinality of $\overline{\mathsf{M}p} : V \leftrightarrow \mathbf{1}$, since this vector describes a single minimum vertex cover of g.

Finally, we tackle the chromatic number problem. Our relation-algebraic solution is the elaboration of a remark made in [9], page 347. Its first step is the column-wise description of all maximal independent sets of g. We start with the fact that if $Y \in 2^V$ is a maximal independent set of g, then for all $x \notin Y$ there exists $y \in Y$ such that R_{xy}. That is, the complement of Y is contained in $\{x \in V \mid \exists\, y \in Y : R_{xy}\}$. As already shown, the converse inclusion characterizes independence of Y. Hence, the maximal independent sets of g, also known as its *kernels* [14], coincide with the fixed points of

$$F : 2^V \to 2^V \qquad F(Y) = \{x \in V \mid \neg\exists\, y \in Y : R_{xy}\}\,. \tag{7}$$

It is an easy exercise to verify that F is modeled column-wise by the relational function $f(v) = \overline{Rv}$. This yields $\mathit{FixPts}(\overline{R\mathsf{M}}) : V \leftrightarrow \mathbf{1}$ as vector description of the set $\mathfrak{K} \subseteq 2^E$ of all kernels of g and, hence,

$$\mathit{KernelList}(R) := \mathsf{M}\,\mathrm{inj}(\mathit{FixPts}(\overline{R\mathsf{M}}))^{\mathsf{T}} : V \leftrightarrow \mathfrak{K}$$

as column-wise description of $\mathfrak{K}$.

In the next step we consider the set system $(V, \mathfrak{K})$ and a minimum *set cover* $\mathfrak{S} := \{X_1, \ldots, X_k\} \subseteq \mathfrak{K}$ of it. That is, $V = \bigcup_{i=1}^{k} X_i$ and there is no collection of

less than k sets from $\mathfrak{K}$ with the same property. The existence of a set cover of $(V, \mathfrak{K})$ follows from the fact that each set of $\mathfrak{K}$ is maximal independent, hence each vertex is contained in at least one set from $\mathfrak{K}$ due to the independence of singleton sets.

If we define the sets $Y_j := X_j \setminus \bigcup_{i=1}^{j-1} X_i$, $1 \leq j \leq k$, then the collection $\mathfrak{C} := \{Y_1, \ldots, Y_k\}$ gives rise to a minimum colouring of g. Here is the proof. $\mathfrak{C}$ is a partition of V: property $V = \bigcup_{i=1}^{k} Y_i$ is obvious, the same holds for $Y_i \cap Y_j = \emptyset$ if $i \neq j$, and, finally, if a set Y_j would be empty, then this would imply $X_j \subseteq \bigcup_{i=1}^{j-1} X_i$, which contradicts that $\mathfrak{S}$ is a minimum set cover. $\mathfrak{C}$ defines a colouring: for all $j, 1 \leq j \leq k$, no pair of distinct vertices $x, y \in Y_j$ is connected via an edge, because of the independence of X_j and, hence, its subset Y_j. This colouring is minimum: each colouring of the graph g with less than k colours would lead to a set cover of $(V, \mathfrak{K})$ smaller than $\mathfrak{S}$.

Summing up: the chromatic number of g equals the cardinality of a minimum set cover of $(V, \mathfrak{K})$. Due to this result, our next task is to develop a vector description of the set of all set covers of $(V, \mathfrak{K})$. In doing so, we suppose $S : V \leftrightarrow \mathfrak{K}$ to be the result of the call $KernelList(R)$, i.e., assume the equivalence of S_{xY} and $x \in Y$ for all $x \in V$ and $Y \in \mathfrak{K}$. This allows to proceed for all $\mathfrak{X} \in 2^{\mathfrak{K}}$ as follows:

$$
\begin{aligned}
V = \textstyle\bigcup_{Y \in \mathfrak{X}} Y &\iff \forall\, x \in V : \exists\, Y \in \mathfrak{X} : x \in Y \\
&\iff \forall\, x : \exists\, Y : \mathsf{M}_{Y\mathfrak{X}} \wedge S_{xY} && \mathsf{M} : \mathfrak{K} \leftrightarrow 2^{\mathfrak{K}} \\
&\iff \forall\, x : (S\mathsf{M})_{x\mathfrak{X}} \\
&\iff \neg\exists\, x : \overline{S\mathsf{M}}^{\mathsf{T}}_{\mathfrak{X}x} \wedge \mathsf{L}_x && \mathsf{L} : V \leftrightarrow \mathbf{1} \\
&\iff (\overline{\overline{S\mathsf{M}}^{\mathsf{T}}\mathsf{L}})_{\mathfrak{X}}\,.
\end{aligned}
$$

From the last expression we get $\overline{\mathsf{L}^{\mathsf{T}}\overline{S\mathsf{M}}}^{\mathsf{T}} : 2^{\mathfrak{K}} \leftrightarrow \mathbf{1}$ as vector description of the set of all set covers of $(V, \mathfrak{K})$, if we apply again the technique of Section 3.2 to avoid the transposition of $\overline{S\mathsf{M}}$. For getting the chromatic number of g it remains, finally, to compute the minimum set covers via

$$MinSetcover(S) := GreEl(\mathsf{C}^{\mathsf{T}}, \overline{\mathsf{L}^{\mathsf{T}}\overline{S\mathsf{M}}}^{\mathsf{T}}) : 2^{\mathfrak{K}} \leftrightarrow \mathbf{1}$$

(C is the size-comparison relation on $2^{\mathfrak{K}}$), to select a point p contained in this vector, and to count the number of *true*-entries of the vector $\mathsf{M}p : \mathfrak{K} \leftrightarrow \mathbf{1}$.

If $\mathfrak{S}$ is the subset of $\mathfrak{K}$ described by $\mathsf{M}p$, then the relational composition $S\,\mathrm{inj}(\mathsf{M}p)^{\mathsf{T}} : V \leftrightarrow \mathfrak{S}$ describes $\mathfrak{S}$ column-wise. That is, the columns of this relation constitute a minimum set cover of $(V, \mathfrak{K})$. As shown above, a column-wise description of a minimum colouring is obtained by scanning $S\,\mathrm{inj}(\mathsf{M}p)^{\mathsf{T}}$

row by row, thereby changing in row j all those *true*-entries to *false*, which already occur in a row $i < j$.

5 Implementation and Experimental Results

RelView [2,1] is a visual computer system for the manipulation of relations and relational programming. The system uses a sophisticated and very efficient implementation of relations via ROBDDs, which has been developed in the course of the Ph.D theses [11] and [13]. This allows also very large relations in computations, especially the membership- and size-comparison-relations, which form the base of our developments in Section 3 and 4.

The user of RelView can manipulate and analyze relations by pre-defined operations and tests and depict the results of computations as Boolean matrices or graphs, where various labeling mechanisms are available in addition. All relational constants and operations we have mentioned in the previous sections are available in the language of RelView. This functionality can be accessed through command buttons and simple mouse-clicks. But the usual way is to combine them into relation-algebraic expressions and user-defined relational functions and programs, respectively.

Each of the relational functions we have presented so far easily can be translated into the language of RelView. We give three examples. The main function *FixPts* of Section 3.2 becomes the following RelView-program:

```
FixPts(Q)
  DECL L, M
  BEG  L = Ln1(Q);
       M = epsi(L)
       RETURN -(L^ * ((M & -Q) | (Q & -M)))^
  END.
```

A formulation of the function *MaxClique* for determining the vector description of all maximum cliques in RelView-syntax looks as follows:

```
MaxClique(R)
  DECL I, M, C
  BEG  I = I(R);
       M = epsi(O(R));
       C = cardrel(O(R))
       RETURN GreEl(C,FixPts(M & -(-(R | I) * M)))
  END.
```

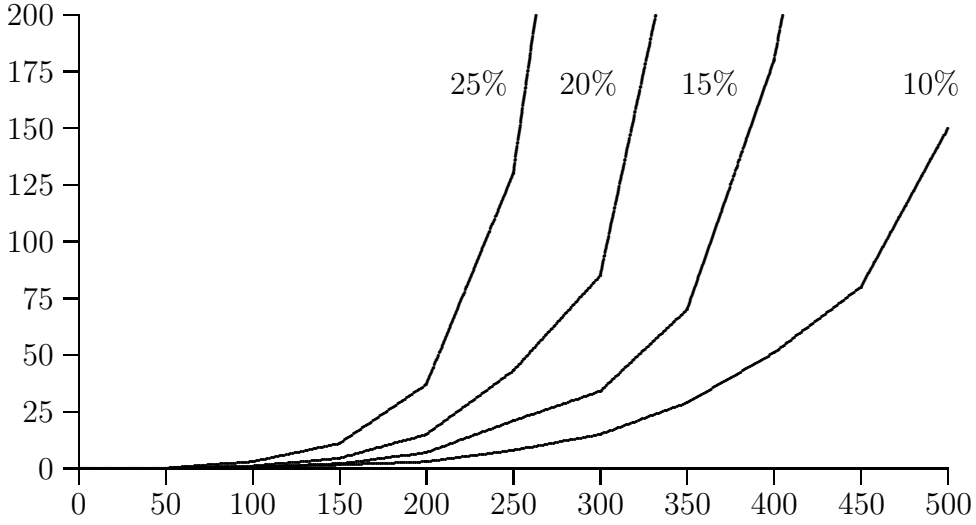
Finally, the computation of a single maximum clique is done by executing the following RelView-program:

```
SingleMaxClique(R)
  DECL M, p
  BEG  M = epsi(O(R));
       p = point(MaxClique(R))
       RETURN M * p
  END.
```

In these programs the operations `O` and `I` compute an empty relation and an identity relation of the same type as their arguments, respectively, `Ln1` yields a universal vector the domain of which is the domain of the argument and the range of which is **1**, `point` selects a point contained in a non-empty vector, and `epsi` and `cardrel` determine the membership-relation $\mathsf{M} : X \leftrightarrow 2^X$ and the size-comparison-relation $\mathsf{C} : 2^X \leftrightarrow 2^X$, respectively, if the argument is a vector with domain X. All these operations are pre-defined.

RelView allows generating uniform random relations, also of specific kind like circuit-free random relations or random permutations (i.e., injective and surjective mappings). The underlying algorithms are described in detail in [13]. It is additionally possible to specify the random relations' densities, in Boolean matrix terminology: the probability of a single entry to be *true*. Based on this feature, a lot of experiments have been carried out with random relations of various sizes and with various densities on a Sun-Fire 880 workstation running Solaris 9 at 750 MHz and with 32 Gbyte of main memory. Apart from computing permanents and minimum colourings, small inputs turned out to be harmless. But, as already mentioned in the introduction, the experiments also showed that our approach often works sufficiently well even in the case of inputs of medium size due to the ROBDD-implementation of relational algebra.

Especially, this holds for the computation of maximum cliques and maximum independent sets. To prove this fact, the following figure shows some experimental results for randomly generated symmetric and irreflexive adjacency relations. In it, the number N of vertices of the random graph is listed on the x-axis and the time (in seconds) needed to execute `MaxClique` is listed on the y-axis. The four curves have been obtained by varying N from 50 to 500 by steps of 50 vertices and the probability of a pair $\{x, y\}$ to be an edge of a graph from 10% to 25% by steps of 5%. We performed at least 20 experiments for each N and each density and computed in all cases the arithmetic mean of the execution times.



Because of lack of memory, in the case of 25% density we have not been able to deal with $N > 300$ vertices. If, however, we restricted us to more sparse graphs, larger numbers of vertices could be treated successfully. To give an impression of the behaviour of `MaxClique`, $N = 700$ and a density of 10% led to approximately 12-15 minutes execution time and, for the same density, $N = 900$ led to roughly 60-80 minutes.

The RelView-program implementing the relational function *MaxInd* of Section 4.3 behaves just the other way round than program `MaxClique`: the denser the graph the less time and storage is needed for its execution. E.g., computing the 92 solutions of the 8-queens-problem takes 12 seconds (the underlying undirected graph possesses 748 of all possible 2016 edges, i.e., 37%).

As already mentioned, the results of Section 4 allow computing permanents and minimum colourings only for small inputs. Here the problems are caused by the intermediate results. This becomes obvious by the following example. Our experiments have shown that a uniform randomly generated graph with 20 vertices and a density of 20% has 86 kernels in the average. Boolean 20×86 matrices are no serious problem for a RelView-implementation of *MinSetcover*. However, Boolean 30×678 matrices are, and such matrices appeared in the average for graphs with 30 vertices and 20% density during our experiments.

6 Conclusion

We have used relational algebra for describing sets and sets of sets and have shown how to compute all fixed points of functions on powersets if they are modeled column-wise by their relational counterparts. Some applications from

different problem domains have been presented, including results of the practical experiments with the RelView system.

The key step of our method is the calculation of the body of the relational function f from the definition of F. Experience with numerous examples has taught us that in many cases the derivation of a relation-algebraic description from a logical specification is of acceptable complexity for people with background in formal logic, relational algebra, and calculational methods. Of course, there are situations where this process requires a lot of creativity, especially if direct products and projections have to be introduced. From a theoretical point of view their use may be unavoidable since the expressive power of relational algebra without products is less than that of first-order logic. But, fortunately, these cases are rare in practice.

Besides the applications described in the paper, we have investigated some further problems. For instance, we have been able to enumerate the vertex sets of all elementary circuits of a directed graph using that they form the minimal successor-closed sets. Among other things, this allows solving the well-known feedback-vertex-problem – at present, because of the very large ROBDD of the set inclusion relation, however only for small graphs. Another example is the computation of all minimum absorbant sets of a directed graph. This helps solving a problem that is dual to the well-known 8-queens-problem in the same sense as the independence/stability condition $v = v \cap \overline{Rv}$ of a vector is dual to its absorptiveness $v = v \cup \overline{Rv}$; see also [14]. The question is: how can the entire chess board be threatened by as few queens as possible and how many solutions do exist? RelView needs only some seconds to compute the answer: 5 queens suffice and there are exactly 4860 different possibilities to place them so that in each case the entire chess board is threatened.

Presently we apply relational algebra and the RelView system for model checking of modal logics. Here fixed points are to compute e.g., when evaluating modal formulae the meaning of which is described with the help of them or when computing closures. The later task appears in the so-called correspondence theory. There are cases where the relation-algebraic properties satisfied by an accessibility relation lead to a closure system $\mathcal{C}$ the induced closure operator $R \mapsto \bigcap\{S \in \mathcal{C} \mid R \subseteq S\}$ of which can be specified via fixed points.

Besides this novel use of RelView, our future work mainly concentrates on two domains. First, we want to re-investigate the problems for which we know that the present solutions use intermediate results with very large ROBDDs. Our hope is to find solutions which avoid this drawback and are applicable also in the case of sets of medium size. The second domain is motivated by the so-called *lectic enumeration* of formal concepts in [6] (which, in the special case of a partial order, corresponds to the enumeration of all Dedekind cuts in

a specific lexicographic order). The algorithm of [6] only uses that $F(Y) = Y^{\uparrow\downarrow}$ is a closure operator and, indeed, constitutes an algorithm that determines all fixed points of such specific functions (see e.g., [7]). A relational formulation of the algorithm is straightforward. Apart from very bad cases it works much more efficient than the method of Section 4.1. Our hope is to find also for other specific classes of functions on powersets more efficient relation-algebraic methods for computing all fixed points. First candidates are antitone functions F, since in this case the search space can be reduced to the interval between the least and greatest fixed point of $F \circ F$.

Acknowledgement. I wish to thank the referees for carefully reading the paper and their helpful comments and suggestions.

References

[1] R. Behnke, R. Berghammer, E. Meyer, F. Schneider, RelView – A system for calculation with relations and relational programming, in: E. Astesiano (Ed.), Proc. 1st Conf. *Fundamental Approaches to Software Engineering*, LNCS 1382, Springer, Berlin (1998), 318-321.

[2] R. Berghammer, B. von Karger, C. Ulke, Relation-algebraic analysis of Petri nets with RelView, in: T. Margaria, B. Steffen (eds.), Proc. 2nd Int. Workshop *Tools and Applications for the Construction and Analysis of Systems*, LNCS 1055, Springer, Berlin (1996), 49-69.

[3] R. Berghammer, B. Leoniuk, U. Milanese, Implementation of relational algebra using binary decision diagrams, in: H. de Swart (Ed.), Proc. 6th Int. Workshop *Relational Methods in Computer Science*, LNCS 2561, Springer, Berlin (2002) 241-257.

[4] J. Desharnais, B. Möller, Least reflexive points of relations. Report 2003-13, Inst. für Inf., Univ. Augsburg (2003) To appear in Higher Order and Symb. Comp.

[5] T. Fujimoto, An extension of Tarski's fixed point theorem and its application to isotone complementarity problems, Math. Programming 28 (1984), 116-118.

[6] B. Ganter, Two basic algorithms in concept analysis, FB 4 Preprint No. 831, Techn. Hochschule Darmstadt (1984).

[7] B. Ganter, Formal concept analysis: algorithmic aspects, Lecture notes, Summer semester 2002, Inst. für Algebra, Techn. Univ. Dresden (2002).

[8] P. Hitchcock, D. Park, Induction rules and proof of termination, in: M. Nivat (Ed.), Automata, Languages and Programming, North-Holland, Amsterdam (1973), 225-251.

[9] T. Emden-Weinert et al., Introduction to graphs and algorithms (in German), Lecture notes, Inst. für Informatik, Humboldt Univ. Berlin, available vie URL www.informatik.hu-berlin.de/Institut/struktur/algorithmen/ga (1996).

[10] M.R. Jerrum, J.A. Sinclair, Approximating the permanent, SIAM J. Comput. 18 (1989), 1149-1178.

[11] B. Leoniuk, ROBDD-based implementation of relational algebra with applications (in German), Ph.D. thesis, Inst. für Inf. u. Prak. Math., Univ. Kiel (2001).

[12] G. Markowsky, Chain-complete posets and directed sets with applications, Algebra Univ. 6 (1976), 53-65.

[13] U. Milanese, On the implementation of a ROBDD-based tool for the manipulation and visualization of relations (in German), Ph.D. thesis, Inst. für Inf. u. Prak. Math., Univ. Kiel (2003).

[14] G. Schmidt, T. Ströhlein, Relations and graphs. Discrete mathematics for computer scientists, EATCS Monographs on Theoretical Computer Science, Springer, Berlin, 1993.

[15] J.A. Sinclair, Algorithms for random generation and counting, Birkhäuser, Boston, 1993.

[16] A. Tarski, On the calculus of relations, J. Symbolic Logic 6 (1841), 73-89.

[17] A. Tarski, A lattice-theoretical fixed point theorem and its applications, Pacific J. Math. 5 (1955), 285-309.