Applying relational algebra and RelView to coalition formation

Rudolf Berghammer, Harrie de Swart, Agnieszka Rusinowska

Applying Relational Algebra and RelView to Coalition Formation*

Rudolf Berghammer[1], Agnieszka Rusinowska[2,3] and Harrie de Swart[4]

[1] Institute of Computer Science, University of Kiel
Olshausenstraße 40, 24098 Kiel, Germany
[2] Nijmegen School of Management, Radboud University Nijmegen
P.O. Box 9108, 6500 HK Nijmegen, The Netherlands
[3] Department of Mathematical Economics, Warsaw School of Economics
Al. Niepodleglosci 162, 02-554 Warsaw, Poland
[4] Department of Philosophy, Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands

Abstract. We present an application of relational algebra to coalition formation. This leads to specifications, which can be executed with the help of the RelView tool after a simple translation into the tool's programming language. As an example we consider a simplification of the situation in Poland after the 2001 elections.

Keywords: RelView, relational algebra, coalition formation, feasible government, dominance, stable government.
Corresponding author: Agnieszka Rusinowska.

1 Introduction

During the last decades a lot of coalition formation models have been proposed and investigated; see e.g., van Deemen [7] or de Vries [15] for an overview. In the present paper we focus on the model, recently introduced in Rusinowska et al. [12]. In this model, the notion of a feasible stable government is central. Roughly speaking, a feasible government is a pair consisting of a majority coalition of parties and a policy supported by this coalition. Stability of a feasible government means that it is not dominated by another feasible one. However, determining feasible stable governments can become quite complex and requires a lot of computations. This holds, in particular, if one is interested in all such governments, for example, in order to choose one of them by means of a bargaining procedure. Hence, using a computer program to calculate which governments are feasible and stable, would be extremely useful for real life applications of the model.

Since some decades relational algebra is used successfully for formal problem specification, prototyping, and algorithm development. See e.g., Schmidt et al.

* Co-operation for this paper was supported by European COST Action 274 "Theory and Applications of Relational Structures as Knowledge Instruments" (TARSKI). We are grateful to two anonymous referees for some useful suggestions.

[13], Brink et al. [6], de Swart el al. [14]. Relations are well suited for modeling and reasoning about many discrete structures (like graphs, games, Petri nets, lattices) and, due to the easy mechanization (using, for instance, Boolean matrices) also for computations on them. Feasibility of a government can be described by two relations which state whether a party accepts a coalition and whether a party supports a policy. The same holds for stability. It can be defined in terms of the 'is-part-of' relations between parties and governments, the dominance relation on governments, and a list of relations comparing governments with respect to the utility of parties. Hence, relational algebra seems to be very promising for computer-aided investigations of coalition formation.

The purpose of this paper is to prove this point. We combine relational algebra and RelView, a tool for the visualization and manipulation of relations and for prototyping and relational programming, to compute the set of all feasible stable governments in the sense of Rusinowska et al. [12]. To illustrate the power of the approach, we solve an example based on the real structure of the Polish government after the 2001 elections.

The remainder of the paper is organized as follows. Section 2 introduces our mathematical model of coalition formation. In Section 3 we collect the facts on relations which are necessary to treat coalition formation relation-algebraically. The treatment itself is done in Section 4 and essentially consists of the specification of the decisive notions of feasibility, dominance, and stability for governments by relation-algebraic expressions. In Section 5 we show how to translate the specifications of Section 4 into RelView code and solve the Polish government example. Finally, we present some concluding remarks in Section 6.

2 The Mathematical Model of Coalition Formation

In this section, we briefly recall some of the main ideas of the model of coalition formation presented in Rusinowska et al. [12], i.e., the notions essential for the application of relational algebra and RelView to the model. Let N be the set of political parties, and P be the set of all policies. A set of parties, i.e., an element of the powerset 2^N, is called a *coalition*. We define a *government* as a pair consisting of a coalition and a policy. Hence,

$$G := \{ (S, p) \mid S \in 2^N \wedge p \in P \}$$

denotes the set of all governments. Usually, we assume that only a majority coalition (i.e., a coalition with more than half of the total number of seats in Parliament) can form a government. Nevertheless, one may easily imagine a government formed by a minority coalition.

Each party is assumed to have preferences on all policies and on all coalitions. A coalition is called *feasible* if it is acceptable to all its members. A policy is *feasible for a given coalition* if it is acceptable to all members of that coalition. A government is said to be *feasible* if it consists of a feasible coalition and a policy feasible for that coalition. By G^* we denote the set of all feasible governments:

$$G^* := \{ g \in G \mid G \text{ is feasible} \}.$$

For each $i \in N$, we assume a utility function $U^{(i)} : G \to \mathbb{R}$, where $U^{(i)}(g)$ denotes the utility (value) of government $g \in G$ to party $i \in N$. A precise definition of the utility of a government to a party has been given in Rusinowska et al. [12]. In Roubens et al. [10], the MacBeth technique has been applied to determine these utilities.

A feasible government $g = (S, p) \in G^*$ *dominates* a feasible government $h \in G^*$ (denoted as $g \succ h$) if the property

$$(\forall\, i \in S : U^{(i)}(g) \geq U^{(i)}(h)) \wedge (\exists\, i \in S : U^{(i)}(g) > U^{(i)}(h)) \tag{1}$$

holds. A feasible government is said to be *stable* if it is dominated by no feasible government. By

$$SG^* := \{g \in G^* \mid \neg\, \exists\, h \in G^* : h \succ g\}$$

we denote the set of all (feasible) stable governments.

3 Relational Preliminaries

In this section, we first recall the basics of relational algebra. Next, we introduce some further relational constructions which are used in the remainder of the paper. For more details on relations and relational algebra, see e.g., Schmidt et al. [13] or Brink et al. [6].

3.1 Relational Algebra

If X and Y are sets, then a subset R of the Cartesian product $X \times Y$ is called a (binary) 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 one of the graphical representations of relations within the RelView tool. Therefore, in this paper we often use Boolean matrix terminology and notation. In particular, we write $R_{x,y}$ instead of $\langle x, y\rangle \in R$.

We assume the reader to be familiar with the basic operations on relations, viz. R^T (*transposition*), $\overline{R}$ (*complement*), $R \cup S$ (*union*), $R \cap S$ (*intersection*), $R; S$ (*composition*), R^* (*reflexive-transitive closure*), and the special relations O (*empty relation*), L (*universal relation*), and I (*identity relation*). If R is included in S we write $R \subseteq S$ and equality of R and S is denoted as $R = S$.

3.2 Modeling of Sets

Relational algebra offers some simple and elegant ways to describe subsets of a given set or, equivalently, predicates on this set. In this paper we will use vectors and membership-relations for this task.

A *vector* v is a relation v with $v = v;\mathsf{L}$. For v being of type $[X \leftrightarrow Y]$ 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 $(v;\mathsf{L})_{x,z}$ 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 $v_{x,\bot}$.

Analogously to linear algebra we will use lower-case letters to denote vectors. 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.

As a second way to model sets we will use the relation-level equivalents of the set-theoretic symbol "$\in$", i.e., *membership-relations* $\varepsilon : X \leftrightarrow 2^X$. These specific relations are defined by $\varepsilon_{x,Y}$ if and only if $x \in Y$, for all $x \in X$ and $Y \in 2^X$. A Boolean matrix representation of ε requires exponential space. However, in Berghammer et al. [4] an implementation of ε using reduced ordered binary decision diagrams (ROBDDs) is presented, the number of vertices of which is linear in the size of X.

3.3 Cartesian Products and Disjoint Unions

Given a Cartesian product $X \times Y$ of two sets X and Y, there are two projection functions which decompose a pair $u = \langle u_1, u_2 \rangle$ into its first component u_1 and its second component u_2. For a relation-algebraic approach it is useful to consider instead of these functions the corresponding *projection relations* $\pi : X \times Y \leftrightarrow X$ and $\rho : X \times Y \leftrightarrow Y$ such that for all $u \in X \times Y$, $x \in X$, and $y \in Y$ we have $\pi_{u,x}$ if and only if $u_1 = x$ and $\rho_{u,y}$ if and only if $u_2 = y$. Projection relations enable us to describe the well-known pairing operation of functional programming relation-algebraically as follows: For relations $R : Z \leftrightarrow X$ and $S : Z \leftrightarrow Y$ we define their *pairing* (frequently also called *fork* or *tupling*) $[R, S] : Z \leftrightarrow X \times Y$ by

$$[R, S] := R;\pi^{\mathsf{T}} \cap S;\rho^{\mathsf{T}}. \tag{2}$$

Using (2), for all $z \in Z$ and $u \in X \times Y$ a simple reflection shows that $[R, S]_{z,u}$ if and only if R_{z,u_1} and S_{z,u_2}. As a consequence, the *exchange relation*

$$E := [\rho, \pi] = \rho;\pi^{\mathsf{T}} \cap \pi;\rho^{\mathsf{T}} \tag{3}$$

of type $[X \times Y \leftrightarrow X \times Y]$ exchanges the components of a pair. This means that for all $u, v \in X \times Y$ the relationship $E_{u,v}$ holds if and only if $u_1 = v_2$ and $u_2 = v_1$.

Analogously to the product the disjoint union $X + Y := (X \times \{1\}) \cup (Y \times \{2\})$ of two sets X and Y leads to the two *injection relations* $\imath : X \leftrightarrow X + Y$ and $\kappa : Y \leftrightarrow X + Y$ such that for all $u \in X + Y$, $x \in X$, and $y \in Y$ we have $\imath_{x,u}$ if and only if $u = \langle x, 1 \rangle$ and $\kappa_{y,u}$ if and only if $u = \langle y, 2 \rangle$. In this case the counterpart of pairing is the *sum* $R + S : X + Y \leftrightarrow Z$ of two relations $R : X \leftrightarrow Z$ and $S : Y \leftrightarrow Z$, defined by

$$R + S := (\imath^{\mathsf{T}};R) \cup (\kappa^{\mathsf{T}};S)\,. \tag{4}$$

From specification (4) we obtain for all $u \in X + Y$ and $z \in Z$ that $(R + S)_{u,z}$ if and only if there exists $x \in X$ such that $u = \langle x, 1\rangle$ and $R_{x,z}$ or there exists $y \in Y$ such that $u = \langle y, 2\rangle$ and $S_{y,z}$.

We end this section with an application of the constructions introduced so far, which is used in the remainder of the paper. Our starting point is the representation of a relation $R : X \leftrightarrow Y$ by a vector $vec(R) : X \times Y \leftrightarrow \mathbf{1}$, which means that for all $x \in X$ and $y \in Y$ the properties $R_{x,y}$ and $vec(R)_{\langle x,y\rangle,\perp}$, or $vec(R)_{\langle x,y\rangle}$ for short, are equivalent. To obtain a relation-algebraic specification of $vec(R)$, i.e., an expression which does not use element relationships (in our notation: does not use indices) but only the constants and operations of relational algebra, we assume $x \in X$ and $y \in Y$ and calculate as follows:

$$\begin{array}{lll} R_{x,y} & \iff \exists\, a : \pi_{\langle x,y\rangle,a} \wedge R_{a,y} & \pi : X \times Y \leftrightarrow X \text{ projection} \\ & \iff (\pi; R)_{\langle x,y\rangle,y} & \\ & \iff \exists\, b : (\pi; R)_{\langle x,y\rangle,b} \wedge \rho_{\langle x,y\rangle,b} & \rho : X \times Y \leftrightarrow Y \text{ projection} \\ & \iff \exists\, b : (\pi; R \cap \rho)_{\langle x,y\rangle,b} \wedge \mathsf{L}_b & \mathsf{L} : Y \leftrightarrow \mathbf{1} \\ & \iff ((\pi; R \cap \rho); \mathsf{L})_{\langle x,y\rangle}\,. & \end{array}$$

An immediate consequence of the last expression of this calculation and the equality of relations is the relation-algebraic specification

$$vec(R) = (\pi; R \cap \rho); \mathsf{L} \tag{5}$$

of the vector $vec(R) : X \times Y \leftrightarrow \mathbf{1}$ (see also Schmidt et al. [13]). Finally, we consider a list $R^{(1)}, R^{(2)}, \ldots, R^{(n)}$ of relations $R^{(i)} : X \leftrightarrow Y$. Let $N := \{1, \ldots, n\}$. If we identify this set with the disjoint union of n copies of $\mathbf{1}$, then

$$C := vec(R^{(1)})^{\mathsf{T}} + \ldots + vec(R^{(n)})^{\mathsf{T}} \tag{6}$$

defines a relation of type $[N \leftrightarrow X \times Y]$ such that, using Boolean matrix terminology, for all $i \in N$ the i^{th} row of C equals the transpose of the vector $vec(R^{(i)})$. Hence, from the above considerations we obtain for all $i \in N, x \in X$, and $y \in Y$ the equivalence of $C_{i,\langle x,y\rangle}$ and $R^{(i)}_{x,y}$.

4 Relational Description of Coalition Formation

Based on the mathematical model of coalition formation in Section 2, in this section we develop relation-algebraic specifications of feasible governments, the dominance relationship, and stable governments. As we will demonstrate in Section 5, these can be translated immediately into the programming language of the RelView tool and then applied to deal with concrete examples.

4.1 Feasibility

In order to develop a relation-algebraic specification of feasible governments, we need two 'acceptability' relations A and B. We assume $A : N \leftrightarrow P$ such that

$$A_{i,p} \iff \text{party } i \text{ accepts policy } p$$

for all $i \in N$ and $p \in P$, and $B : N \leftrightarrow 2^N$ such that

$$B_{i,S} \iff \text{party } i \text{ accepts coalition } S$$

for all $i \in N$ and $S \in 2^N$.

Based on the two acceptability relations A and B and the description of feasibility given in Section 2, next we consider the following three relations:

1. A relation $\mathit{isFea}(A) : 2^N \leftrightarrow P$ such that a coalition $S \in 2^N$ and a policy $p \in P$ are in relationship $\mathit{isFea}(A)_{S,p}$ if and only if p is feasible for S. A formal predicate logic definition of this is
$$\mathit{isFea}(A)_{S,p} \iff \forall\, i : i \in S \to A_{i,p}\,. \tag{7}$$
2. A vector $\mathit{feaC}(B) : 2^N \leftrightarrow \mathbf{1}$ which describes the set of all feasible coalitions. Here we have for all $S \in 2^N$ the predicate logic definition
$$\mathit{feaC}(B)_S \iff \forall\, i : i \in S \to B_{i,S}\,. \tag{8}$$
3. A relation $\mathit{feaG}(A,B) : 2^N \leftrightarrow P$ which coincides with the set G^* of feasible governments. Using the relations introduced so far, hence we have for all coalitions $S \in 2^N$ and policies $p \in P$ the description
$$\mathit{feaG}(A,B)_{S,p} \iff \mathit{isFea}(A)_{S,p} \;\wedge\; \mathit{feaC}(B)_S\,. \tag{9}$$

Our goal is to obtain from the predicate logic definitions (7), (8), and (9) of the relations $\mathit{isFea}(A)$, $\mathit{feaC}(B)$, and $\mathit{feaG}(A,B)$ equivalent relation-algebraic specifications. The proof of the following theorem shows how to calculate these. Besides the predicate logic definitions and some simple logical laws it only uses well-known correspondences between certain logical constructions and relation-algebraic operations (see e.g., Schmidt et al. [13]).

Theorem 4.1 *Assume $\varepsilon : N \leftrightarrow 2^N$ to be the membership-relation on the set of parties. Then we have*

$$\mathit{isFea}(A) = \overline{\varepsilon^{\mathsf{T}}; \overline{A}}\,, \tag{10}$$

$$\mathit{feaC}(B) = \overline{(\varepsilon \cap \overline{B})^{\mathsf{T}}; \mathsf{L}}\,, \tag{11}$$

$$\mathit{feaG}(A,B) = \overline{\varepsilon^{\mathsf{T}}; \overline{A}} \cap \overline{(\varepsilon \cap \overline{B})^{\mathsf{T}}; \mathsf{L}; \mathsf{L}}\,. \tag{12}$$

Proof: For all $S \in 2^N$ and $p \in P$ we are able to calculate

$$\begin{aligned}
\mathit{isFea}(A)_{S,p} &\iff \forall\, i : i \in S \to A_{i,p} && (7)\\
&\iff \neg\, \exists\, i : i \in S \wedge \overline{A}_{i,p}\\
&\iff \neg\, \exists\, i : \varepsilon^{\mathsf{T}}_{S,i} \wedge \overline{A}_{i,p}\\
&\iff (\overline{\varepsilon^{\mathsf{T}}; \overline{A}})_{S,p}\,,
\end{aligned}$$

and from the last expression of this derivation we get (10) due to the definition of equality of relations. In the same way (11) follows from

$$\begin{array}{lll} feaC(B)_S & \Longleftrightarrow & \forall\, i : i \in S \rightarrow B_{i,S} \qquad (8) \\ & \Longleftrightarrow & \neg\, \exists\, i : i \in S \wedge \overline{B}_{i,S} \\ & \Longleftrightarrow & \neg\, \exists\, i : \varepsilon_{i,S} \wedge \overline{B}_{i,S} \\ & \Longleftrightarrow & \neg\, \exists\, i : (\varepsilon \cap \overline{B})_{i,S} \\ & \Longleftrightarrow & \neg\, \exists\, i : (\varepsilon \cap \overline{B})^{\mathsf{T}}_{S,i} \wedge \mathsf{L}_i \qquad \mathsf{L} : P \leftrightarrow \mathbf{1} \\ & \Longleftrightarrow & \overline{((\varepsilon \cap \overline{B})^{\mathsf{T}}; \mathsf{L})_S} \end{array}$$

for all $S \in 2^N$ and the equality of relations, and (12) follows from

$$\begin{array}{lll} feaG(A,B)_{S,p} & \Longleftrightarrow & isFea(A)_{S,p} \wedge feaC(B)_S \qquad (9) \\ & \Longleftrightarrow & isFea(A)_{S,p} \wedge (\exists\, i : feaC(B)_{S,i} \wedge \mathsf{L}_{i,p}) \qquad \mathsf{L} : N \leftrightarrow P \\ & \Longleftrightarrow & isFea(A)_{S,p} \wedge (feaC(B); \mathsf{L})_{S,p} \\ & \Longleftrightarrow & (isFea(A) \cap feaC(B); \mathsf{L})_{S,p} \\ & \Longleftrightarrow & (\overline{\varepsilon^{\mathsf{T}}; \overline{A}} \cap \overline{(\varepsilon \cap \overline{B})^{\mathsf{T}}; \mathsf{L}; \mathsf{L}})_{S,p} \qquad (10), (11) \end{array}$$

for all $S \in 2^N$ and $p \in P$ and the equality of relations. □

It should be remarked that the relation-algebraic specification of $feaG(A,B)$ uses two different universal relations, which are denoted with the same symbol L. The inner L of (12) has type $[P \leftrightarrow \mathbf{1}]$, i.e., is a universal vector, whereas the type of the outer L of (12) is $[N \leftrightarrow P]$, i.e., this L is a 'proper' universal relation.

4.2 Dominance

Now, we consider the set G^* of all feasible governments and want to develop a relation-algebraic specification of the dominance relationship between feasible governments. To this end, we suppose a relational description of government membership to be given, that is, a relation $M : N \leftrightarrow G^*$ such that

$$M_{i,g} \iff \text{party } i \text{ is a member of government } g$$

for all $i \in N$ and $g \in G^*$. Moreover, for each party $i \in N$, we introduce a utility (or comparison) relation $R^{(i)} : G^* \leftrightarrow G^*$ by the definition

$$R^{(i)}_{g,h} \iff U^{(i)}(g) \geq U^{(i)}(h)$$

for all $g, h \in G^*$. Based on these relations, we are able to introduce a global utility (or comparison) relation $C : N \leftrightarrow G^* \times G^*$ as follows: For all $i \in N$ and $g, h \in G^*$ we define $C_{i,\langle g,h\rangle}$ if and only if $R^{(i)}_{g,h}$. If the set N is finite and enumerated, i.e., $N = \{i_1, \ldots, i_n\}$, then this leads to the list $R^{(i_1)}, \ldots, R^{(i_n)}$ of utility relations and using (5) and (6) we immediately get the relation-algebraic specification

$$C = vec(R^{(i_1)})^{\mathsf{T}} + \ldots + vec(R^{(i_n)})^{\mathsf{T}}. \qquad (13)$$

After these preparations, now we consider the dominance relationship. Combining its predicate logic definition in (1) with the relations M and C, we get

$$g \succ h \iff (\forall\, i : M_{i,g} \to C_{i,\langle g,h\rangle}) \wedge (\exists\, i : M_{i,g} \wedge \overline{C}_{i,\langle h,g\rangle}) \tag{14}$$

for all $g, h \in G^*$. Now, we use the exchange relation $E : G^* \times G^* \leftrightarrow G^* \times G^*$ and transform the expression $\overline{C}_{i,\langle h,g\rangle}$ of (14) as follows:

$$\begin{aligned} \overline{C}_{i,\langle h,g\rangle} &\iff \exists\, \langle a,b\rangle : \overline{C}_{i,\langle a,b\rangle} \wedge \langle a,b\rangle = \langle h,g\rangle \\ &\iff \exists\, \langle a,b\rangle : \overline{C}_{i,\langle a,b\rangle} \wedge E_{\langle a,b\rangle,\langle g,h\rangle} \\ &\iff (\overline{C}; E)_{i,\langle g,h\rangle}\,. \end{aligned}$$

This equivalence yields the following description of dominance:

$$g \succ h \iff (\forall\, i : M_{i,g} \to C_{i,\langle g,h\rangle}) \wedge (\exists\, i : M_{i,g} \wedge (\overline{C}; E)_{i,\langle g,h\rangle})\,. \tag{15}$$

The use of pairs $\langle g, h\rangle$ and the quantified i in the right-hand side of (15) suggests to specify the dominance relationship as a vector $\mathit{dominance}(M, C) : G^* \times G^* \leftrightarrow \mathbf{1}$ which describes the set of pairs $u \in G^* \times G^*$ such that the first component dominates the second component. In the proof of the next theorem we show how to obtain from (15) a relation-algebraic specification of this vector.

Theorem 4.2 *Let $\pi : G^* \times G^* \leftrightarrow G^*$ and $\rho : G^* \times G^* \leftrightarrow G^*$ be the projection relations and $E : G^* \times G^* \leftrightarrow G^* \times G^*$ the exchange relation. If we define*

$$\mathit{dominance}(M, C) = \overline{(\pi; M^\mathsf{T} \cap \overline{C}^\mathsf{T}); \mathsf{L}} \cap (\pi; M^\mathsf{T} \cap E; \overline{C}^\mathsf{T}); \mathsf{L}\,, \tag{16}$$

then we have for all $u = \langle g, h\rangle \in G^* \times G^*$ that $\mathit{dominance}(M, C)_u$ if and only if $g \succ h$.

Proof: We start our calculation with the first conjunct of the right-hand side of (15) and transform it as follows:

$$\begin{aligned} \forall\, i : M_{i,g} \to C_{i,\langle g,h\rangle} &\iff \neg\, \exists\, i : M_{i,g} \wedge \overline{C}_{i,\langle g,h\rangle} \\ &\iff \neg\, \exists\, i : M^\mathsf{T}_{g,i} \wedge \overline{C}_{i,u} \\ &\iff \neg\, \exists\, i : (\exists\, g' : \pi_{u,g'} \wedge M^\mathsf{T}_{g',i}) \wedge \overline{C}^\mathsf{T}_{u,i} \\ &\iff \neg\, \exists\, i : (\pi; M^\mathsf{T})_{u,i} \wedge \overline{C}^\mathsf{T}_{u,i} \\ &\iff \neg\, \exists\, i : (\pi; M^\mathsf{T} \cap \overline{C}^\mathsf{T})_{u,i} \wedge \mathsf{L}_i \qquad\qquad \mathsf{L} : N \leftrightarrow \mathbf{1} \\ &\iff \overline{((\pi; M^\mathsf{T} \cap \overline{C}^\mathsf{T}); \mathsf{L})_u}\,. \end{aligned}$$

In a similar way we treat the second conjunct of the right-hand side of (15). Here we can proceed as follows:

$$\begin{aligned} \exists\, i : M_{i,g} \wedge (\overline{C}; E)_{i,\langle g,h\rangle} &\iff \exists\, i : (\pi; M^\mathsf{T})_{u,i} \wedge (\overline{C}; E)_{i,\langle g,h\rangle} && \text{see above} \\ &\iff \exists\, i : (\pi; M^\mathsf{T})_{u,i} \wedge (\overline{C}; E)^\mathsf{T}_{u,i} \wedge \mathsf{L}_i && \mathsf{L} : N \leftrightarrow \mathbf{1} \\ &\iff \exists\, i : (\pi; M^\mathsf{T})_{u,i} \wedge (E; \overline{C}^\mathsf{T})_{u,i} \wedge \mathsf{L}_i && E = E^\mathsf{T} \\ &\iff \exists\, i : (\pi; M^\mathsf{T} \cap E; \overline{C}^\mathsf{T})_{u,i} \wedge \mathsf{L}_i \\ &\iff ((\pi; M^\mathsf{T} \cap E; \overline{C}^\mathsf{T}); \mathsf{L})_u\,. \end{aligned}$$

Combining these derivations with the definition (16) of $dominance(M,C)$ and property (15) yields the desired result. □

During the proof of Theorem 4.2 we have applied the symmetry of the exchange relation E to make a later RelView-implementation of $dominance(M,C)$ more efficient. Symmetry of E is an immediate consequence of its element-wise description. But it also can be shown with the help of the relation-algebraic specification (3) of E and a relation-algebraic axiomatization of the projection relations as, e.g., presented in Schmidt et al. [13].

If one is interested in the dominance relationship between feasible governments as a relation of type $[G^* \leftrightarrow G^*]$ instead of a vector as we described it, then such a relation can be obtained using that $rel(v) = \pi^\mathsf{T}; (\rho \cap (v; \mathsf{L}))$ describes a vector $v : X \times Y \leftrightarrow \mathbf{1}$ as relation $rel(v) : X \leftrightarrow Y$, i.e., $v_{\langle x,y \rangle}$ and $rel(v)_{x,y}$ are equivalent for all $x \in X$ and $y \in Y$ (where π, ρ are the projection relations; see again Schmidt et al. [13]).

4.3 Stability

Finally, we consider stability of feasible governments. Here we start with a stability vector $stable(M,C) : G^* \leftrightarrow \mathbf{1}$ which describes the subset SG^* of G^*, i.e., the subset of all (feasible) stable governments. From the definition of SG^* in Section 2 and Theorem 4.2 we get

$$stable(M,C)_g \iff \neg\, \exists\, h : dominance(M,C)_{\langle h,g \rangle} \tag{17}$$

for all $g \in G^*$. This description immediately yields a relation-algebraic specification of stability.

Theorem 4.3 *Let $\rho : G^* \times G^* \leftrightarrow G^*$ be the second projection relation of the Cartesian product $G^* \times G^*$. Then*

$$stable(M,C) = \overline{\rho^\mathsf{T}; dominance(M,C)}\,. \tag{18}$$

Proof: For all $g \in G^*$ we have

$$\begin{aligned} stable(M,C)_g &\iff \neg\, \exists\, h : dominance(M,C)_{\langle h,g \rangle} && (17)\\ &\iff \neg\, \exists\, u : \rho_{u,g} \wedge dominance(M,C)_u \\ &\iff \neg\, \exists\, u : \rho^\mathsf{T}_{g,u} \wedge dominance(M,C)_u \\ &\iff (\overline{\rho^\mathsf{T}; dominance(M,C)})_g \end{aligned}$$

and, hence, (18) is a consequence of the equality of relations. □

5 Implementation and Example

In order to illustrate our approach, in the following we use the RelView tool to solve a simple theoretical example based on the real structure of the Polish government after the 2001 elections. The extended version of this example, without using RelView, is presented in Rusinowska et al. [11].

5.1 The RelView Tool

Relational algebra has a fixed and surprisingly small set of constants and operations which – in the case of finite carrier sets – can be implemented very efficiently. At Kiel University we have developed a visual computer system for the visualization and manipulation of relations and for relational prototyping and programming, called RelView. The tool is written in the C programming language, uses reduced ordered binary decision diagrams for implementing relations, and makes full use of the X-windows graphical user interface. Details and applications can be found, for instance, in Berghammer et al. [3], Behnke et al. [1], Berghammer et al. [4], and Berghammer et al. [5].

The main purpose of the RelView tool is the evaluation of relation-algebraic expressions. These are constructed from the relations of its workspace using predefined operations and tests, user-defined relational functions, and user-defined relational programs. A relational program is much like a function procedure in the programming languages Pascal or Modula 2, except that it only uses relations as data type. It starts with a head line containing the program name and the list of formal parameters, which stand for relations. Then the declaration of the local relational domains, functions, and variables follows. Domain declarations can be used to introduce projection relations and pairings of relations in the case of Cartesian products, and injection relations and sums of relations in the case of disjoint unions, respectively. The third part of a program is the body, a while-program over relations. As a program computes a value, finally, its last part consists of a return-clause, which is a relation-algebraic expression whose value after the execution of the body is the result.

For example, the relation-algebraic specification $dominance(M, C)$ of the vector describing the dominance relationship between feasible governments immediately leads to the following RelView-program.

```
dominance(M,C)
  DECL Prod = PROD(M^*M,M^*M);
       pi, rho, E
  BEG  pi = p-1(Prod);
       rho = p-2(Prod);
       E = [rho,pi]
       RETURN -dom(pi*M^ & -C^) & dom(pi*M^ & E*-C^)
  END.
```

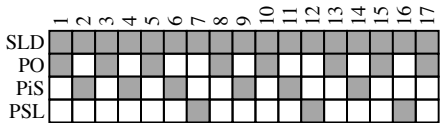
In this program the first declaration introduces `Prod` as a name for the direct product $G^* \times G^*$. Using `Prod`, the projection relations and the exchange relation are then computed by the three assignments of the body and stored as `pi`, `rho`, and `E`, respectively. The return-clause of the program consists of a direct translation of (16) into RelView-notation, where `^`, `-`, `&`, and `*` denote transposition, complement, intersection, and composition, and, furthermore, the pre-defined operation `dom` computes for a relation $R : X \leftrightarrow Y$ the vector $R;\mathsf{L} : X \leftrightarrow \mathbf{1}$.

Also a translation of the remaining relation-algebraic specifications of Section 4 into RelView-code is straightforward.

5.2 The Example

In order to illustrate the application of RelView to coalition formation, we use a theoretical example presented in Rusinowska et al. [11]. It is based on the structure of Polish Parliament after the 2001 elections. The Lower House of Polish Parliament after these elections consists of seven parties: the Alliance of the Democratic Left and Labor Union (SLD), Civil Platform (PO), Self-Defence, Law and Justice (PiS), Polish Peasant Party (PSL), Polish Family Alliance, and German Minority.

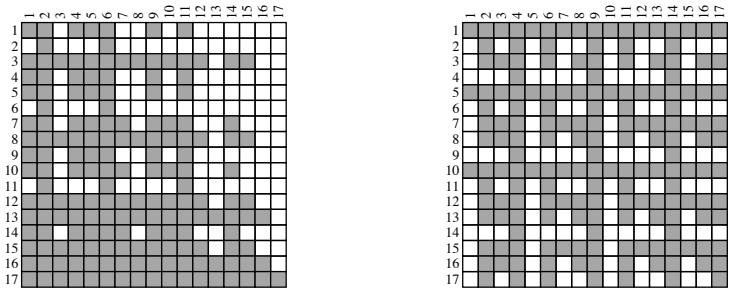
In our example, the acceptability relationships showed that 17 governments are feasible. Only four of the seven parties are involved in these feasible governments, viz. SLD, PO, PiS, and PSL. The following picture shows the membership relation M as Boolean 4×17 matrix as presented by RelView. Here a black square means 1 and a white square means 0 so that, for example, the first government consists of SLD and PO.



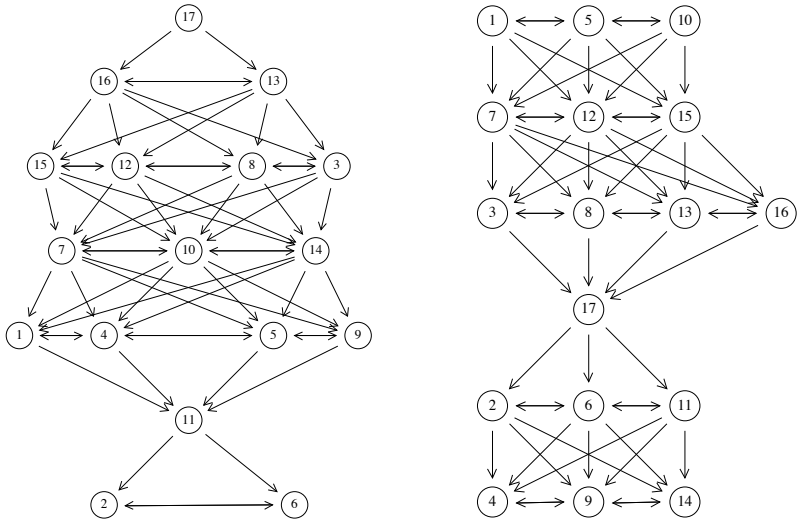
Based on the parties' preferences, in Rusinowska et al. [11] the following utilities of the feasible governments have been determined.

Government	$U^{(\mathrm{SLD})}(g)$	$U^{(\mathrm{PO})}(g)$	$U^{(\mathrm{PiS})}(g)$	$U^{(\mathrm{PSL})}(g)$
$g_1 = (\text{SLD-PO}, p_1)$	$\frac{4}{3}$	$\frac{7}{3}$	$\frac{4}{3}$	$-\frac{7}{3}$
$g_2 = (\text{SLD-PiS}, p_1)$	$\frac{2}{3}$	-1	$\frac{5}{3}$	$-\frac{17}{3}$
$g_3 = (\text{SLD-PO}, p_2)$	2	$\frac{5}{3}$	2	$-\frac{17}{3}$
$g_4 = (\text{SLD-PiS}, p_2)$	$\frac{4}{3}$	$-\frac{5}{3}$	$\frac{7}{3}$	-9
$g_5 = (\text{SLD-PO}, p_3)$	$\frac{4}{3}$	$\frac{7}{3}$	$\frac{4}{3}$	$\frac{4}{3}$
$g_6 = (\text{SLD-PiS}, p_3)$	$\frac{2}{3}$	-1	$\frac{5}{3}$	-2
$g_7 = (\text{SLD-PSL}, p_3)$	$\frac{5}{3}$	2	$\frac{4}{3}$	2
$g_8 = (\text{SLD-PO}, p_4)$	2	$\frac{5}{3}$	2	-2
$g_9 = (\text{SLD-PiS}, p_4)$	$\frac{4}{3}$	$-\frac{5}{3}$	$\frac{7}{3}$	$-\frac{16}{3}$
$g_{10} = (\text{SLD-PO}, p_5)$	$\frac{5}{3}$	$\frac{7}{3}$	$\frac{4}{3}$	$\frac{4}{3}$
$g_{11} = (\text{SLD-PiS}, p_5)$	1	-1	$\frac{5}{3}$	-2
$g_{12} = (\text{SLD-PSL}, p_5)$	2	2	$\frac{4}{3}$	2
$g_{13} = (\text{SLD-PO}, p_6)$	$\frac{7}{3}$	$\frac{5}{3}$	2	-2
$g_{14} = (\text{SLD-PiS}, p_6)$	$\frac{5}{3}$	$-\frac{5}{3}$	$\frac{7}{3}$	$-\frac{16}{3}$
$g_{15} = (\text{SLD-PO}, p_7)$	2	2	$-\frac{8}{3}$	$\frac{2}{3}$
$g_{16} = (\text{SLD-PSL}, p_7)$	$\frac{7}{3}$	$\frac{5}{3}$	$-\frac{8}{3}$	$\frac{4}{3}$
$g_{17} = (\text{SLD-PO}, p_8)$	$\frac{8}{3}$	$\frac{4}{3}$	-2	$-\frac{8}{3}$

With the help of the data presented in this table we can define the utility relations $R^{(i)}$, where $i \in \{\text{SLD, PO, PiS, PSL}\}$. To give an impression how these relations look as RelView-matrices, the next two pictures show the matrices for $R^{(\text{SLD})}$ (left-hand side) and $R^{(\text{PO})}$ (right-hand side).



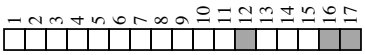
An easy way to obtain these relations is, first to draw – from the table above – the directed graphs expressing the immediate neighborhood relationships of the utilities, and then to compute the reflexive-transitive closures of the graphs' relations. For example, the above matrices are obtained from the following graphs:



At this place we would like to point out that theoretically it is not necessary that the utilities of all governments to the parties are available as real numbers. As the two pictures show, it would suffice to have the relationships at hand which are expressed by the different horizontal layers of the graphs of the relations $R^{(i)}$ and, e.g., obtained by a finite and linearly ordered set of utilities. But the techniques and systems used in practice (e.g., MacBeth in Roubens et al. [10]) usually assume numerical values for all utilities.

Having defined the membership relation and the utility (or comparison) relations, the RelView system is able to compute the global utility relation C

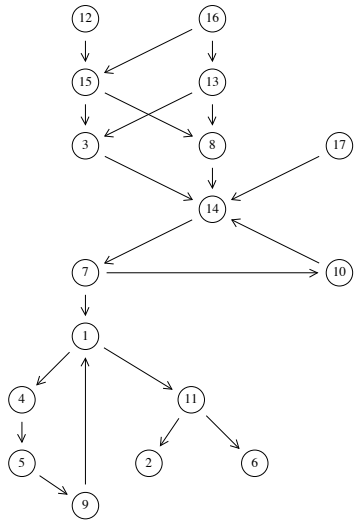
and, after that, the vector describing the set of stable governments. The output of our example, i.e., the vector showing all stable governments, is presented in the following picture. To save place, it shows the vector $stable(M, C)$ in the transposed form, i.e., as Boolean 1×17 matrix.



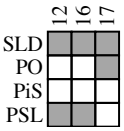
According to this Boolean row vector, 3 out of 17 feasible governments are stable: governments g_{12}, g_{16} and g_{17}. We like to point out that it took only 0.04 second for RelView to determine these stable governments on a Sun-Fire 880 workstation (running Solaris 9 at 750 MHz).

To test the efficiency of our RelView-programs, we have performed further experiments with randomly generated larger examples, e.g., 20 parties and 400 feasible governments. In each case it took at most some seconds to compute all stable governments – although 400 feasible governments lead to a C-matrix with 160 000 columns.

If there are (as in the Polish elections example) only a small number of feasible governments, then a RelView-user immediately obtains the stable governments from their vector description or, as an alternative, by first computing dominance as a relation on $G^* \times G^*$ as sketched in Section 4.2 (applying the operation *rel* to the vector $dominance(M, C)$) and looking after that for the sources (roots) of the graph-representation of this relation. In the latter case, things become even more easy if one considers only the graph of the immediate neighborhood relationships of the dominance relation, that is, a so-called transitive reduction. For the Polish government example, we obtain the following picture (drawn with RelView using the layout algorithm of Gansner et al. [8]), showing again that g_{12}, g_{16} and g_{17} are the stable governments:



In the case of a large number of feasible governments, say 400 as mentioned above, stability hardly can be seen from the vector or the graph description. Nevertheless, there is a very understandable RelView-visualization of the set of all stable governments, viz. as the columns of a relation of type $[M \leftrightarrow SG^*]$. In our example, this relation looks as given in the following picture:



The general method to obtain such a column-wise description of stable governments is to compute the composition $M; id^{\mathsf{T}}$, where $M : N \leftrightarrow G^*$ is the relation describing government membership and $id : SG^* \leftrightarrow G^*$ is the representation of the identical function from SG^* to G^* as a relation. The computation of id with the RelView tool is very easy (and fast), since this relation corresponds to the so-called injective mapping generated by the vector $stable(M, C)$ and RelView possesses a corresponding pre-defined operation `inj` for computing such mappings from vectors. A formal relation-algebraic specification of this operation, e.g., can be found in Berghammer et al. [2].

6 Conclusion

We formulated the notions of feasibility, dominance, and stability for governments in relation-algebraic terms. This enables us to use the RelView tool to compute which governments are stable. Such an application is important if one wants to apply the coalition formation model of Rusinowska et al. [12] to real life situations. We illustrated the use of RelView by a simplified version of the formation of a government in Poland after the 2001 elections.

We feel that our approach has some advantages. First of all, the formality of relational algebra drastically reduces the danger of making errors. As our examples (and many others) show, it is often possible to 'calculate' concise relational algorithms from formal mathematical specifications, so that correctness is guaranteed by construction. Secondly, the application of RelView not only allows a relational algorithm to be executed after its simple translation into the system's programming language. RelView also can be used to visualize its results and to animate its mode of operation. Besides its efficiency (which follows from the fact that the underlying technology is based on ROBDDs), we believe that the real attraction of the tool is its flexibility and the shortness and clearness of its programs (a consequence of relational algebra). New relation-based concepts are introduced all the time – in decision theory as well as in other disciplines – and experience has taught us that RelView is an ideal tool for experimenting and playing with them while avoiding unnecessary overhead. This is demonstrated by many other examples, including also game-theoretic ones (like the computation of all kernels and winning positions if a two-player game is described by a graph; see e.g., von Karger and Berghammer [9] and Berghammer et al. [2]).

For the future we plan to develop relation-algebraic formulations of bargaining procedures, in order to be able to select a specific stable government if more than one stable government exists. We are also interested in applications of relational algebra and the RelView tool to other problems of decision theory, for instance choices in social groups where the decision of group members can be influenced by other ones.

References

1. R. Behnke, R. Berghammer, E. Meyer and P. Schneider, RelView — *A system for calculation with relations and relational programming*, In: E. Astesiano (ed.), Proc. Conf. "Fundamental Approaches to Software Engineering (FASE '98)", LNCS 1382, Springer, 318-321 (1998)
2. R. Berghammer and T. Hoffmann, *Deriving relational programs for computing kernels by reconstructing a proof of Richardson's theorem*, Science of Computer Programming 38, 1-25 (2000)
3. R. Berghammer, B. Karger von and C. Ulke, *Relation-algebraic analysis of Petri nets with* RelView, In: T. Margaria and B. Steffen (eds.), Proc. 2nd Workshop "Tools and Applications for the Construction and Analysis of Systems (TACAS '96)", LNCS 1055, Springer, 49-69 (1996)
4. R. Berghammer, B. Leoniuk and U. Milanese, *Implementation of relational algebra using binary decision diagrams*, In: H. Swart de (ed.), Proc. 6th Int. Workshop "Relational Methods in Computer Science", LNCS 2561, Springer, 241-257 (2002)
5. R. Berghammer, G. Schmidt and M. Winter, RelView *and* Rath – *Two systems for dealing with relations*, In: [14], 1-16 (2003)
6. C. Brink, W. Kahl and G. Schmidt (eds.), *Relational Methods in Computer Science*, Advances in Computing Science, Springer (1997)
7. A. Deemen van, *Coalition formation and social choice*, Kluwer (1997)
8. E.R. Gansner, E. Koutsofios, S.C. North and K.P. Vo, *A technique for drawing directed graphs*, IEEE Trans. Software Eng. 19, 214-230 (1993).
9. B. Karger von and R. Berghammer, *Computing kernels in directed bichromatic graphs*, Information Processing Letters 62, 5-11 (1997)
10. M. Roubens, A. Rusinowska and H. Swart de, *Using MacBeth to determine utilities of governments to parties in coalition formation*, Forthcoming in European Journal of Operational Research (2005)
11. A. Rusinowska and H. Swart de, *Negotiating a stable government - an application of bargaining theory to a coalition formation model*, Submitted (2004)
12. A. Rusinowska, H. Swart de and J.W. Rijt van der, *A new model of coalition formation*, Social Choice and Welfare 24, 129-154 (2005)
13. G. Schmidt and T. Ströhlein, *Relations and Graphs*, Discrete Mathematics for Computer Scientists, EATCS Monographs on Theoret. Comput. Sci., Springer (1993)
14. H. Swart de, E. Orlowska, G. Schmidt and M. Roubens (eds.), *Theory and Applications of Relational Structures as Knowledge Instruments*, LNCS 2929, Springer (2003)
15. M. Vries de, *Covering with your closest neighbour: An assessment of spatial coalition formation theories*, Print Partners Ipskamp (1999)