European Journal of Operational Research 226 (2013) 636–645

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Computing tournament solutions using relation algebra and RelView ☆

Rudolf Berghammer [a], Agnieszka Rusinowska [b,*], Harrie de Swart [c]

[a] Institut für Informatik, Christian-Albrechts-Universität Kiel, Olshausenstraße 40, 24098 Kiel, Germany
[b] Paris School of Economics, CNRS, Centre d'Economie de la Sorbonne, 106-112 Boulevard de l'Hopital, 75647 Paris Cedex 13, France
[c] Faculty of Philosophy, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

ARTICLE INFO

Article history:
Received 5 July 2012
Accepted 18 November 2012
Available online 29 November 2012

Keywords:
Tournament
Relational algebra
RelView
Copeland set
Condorcet non-losers
Schwartz set

ABSTRACT

We describe a simple computing technique for the tournament choice problem. It rests upon relational modeling and uses the BDD-based computer system RelView for the evaluation of the relation-algebraic expressions that specify the solutions and for the visualization of the computed results. The Copeland set can immediately be identified using RelView's labeling feature. Relation-algebraic specifications of the Condorcet non-losers, the Schwartz set, the top cycle, the uncovered set, the minimal covering set, the Banks set, and the tournament equilibrium set are delivered. We present an example of a tournament on a small set of alternatives, for which the above choice sets are computed and visualized via RelView. The technique described in this paper is very flexible and especially appropriate for prototyping and experimentation, and as such very instructive for educational purposes. It can easily be applied to other problems of social choice and game theory.

1. Introduction

Science is a process for obtaining new insights and building new knowledge in the form of testable explanations and predictions about the universe. Systematic experiments are an accepted means for doing science and they become increasingly important as one proceeds in investigations. In natural sciences they are used since centuries. Also in social sciences, which apply scientific methods to study human behavior and social patterns, experimental and empirical methods are of importance. But in the meantime they have also become important in formal sciences, like mathematics and theoretical computer science, for identifying properties and patterns, and for testing and especially falsifying conjectures. In this context, tool support is indispensable. Computer programs are used in numerous scientific fields to calculate results as well as to elucidate the underlying mathematical principles by means of visualization and animation. Frequently use is made of general computer algebra systems, like maple and mathematica. But also systems that focus on specific domains of applications are applied.

RelView (cf. [3,26]) is such a so-called *specific purpose computer algebra system* for (heterogeneous) relation algebra in the sense of [30,31]. More precisely, RelView is a tool for the visualization and manipulation of relations, for prototyping and relational programming, and as such it appears to be very useful and appropriate for applications to social choice and game theory. In this system, computational tasks on relations can be described by short and concise programs which frequently consist of only a few lines that present the relation-algebraic expressions of the notions in question. Such programs are easy to alter in case of slightly changed specifications. Combining this with RelView's possibilities for visualization and stepwise execution of programs makes RelView suitable for experimentation and exploration, while avoiding unnecessary expenditure of work. Another advantage of the system is its implementation of relations via binary decision diagrams (BDDs) that proved to be superior to many other well-known implementations, like Boolean matrices, lists of pairs and lists of successor or predecessor lists. This leads to an amazing computational power, in particular if the solution of a hard problem requires the enumeration of a huge set of 'interesting objects' and a search through it. Applications in this regard can be found, e.g., in [2,3,25,26].

In [4–6] we have combined relation algebra and RelView to solve some problems of computational social choice theory, viz. the formation of stable governments (see [29]) and the determination of the strength and influence of players (see, e.g., [20]). The first problem is a specific instance of one of the most interesting problems of social choice theory, viz. the computation of the set of most desirable alternatives according to an *asymmetric dominance relation* on given alternatives that summarizes the results of the individual preferences. Since the dominance relation may contain cycles, the concept of a *best alternative* that dominates all other alternatives is not applicable in most cases. Even undominated alternatives need not exist. To get around these problems, in the

☆ Co-operation for this paper is supported by European Science Foundation EUROCORES Programme – LogICCC.

* Corresponding author. Tel.: +33 (0)1 4407 8212; fax: +33 (0)1 4407 8301.
E-mail address: agnieszka.rusinowska@univ-paris1.fr (A. Rusinowska).

0377-2217/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2012.11.025

literature so-called *choice sets* are considered that take over the role of the best/undominated alternative(s).

In this paper we show how certain important choice sets can be computed using relation algebra and the RelView tool. We restrict our analysis to complete dominance relations, where each pair of different elements is related. Such relations are known as *tournament relations* and they arise if the pair-wise comparison method and a tie-breaking rule are used for preference aggregation. The elements of the choice sets are called *tournament winners*. But most of our results also hold in the case of non-complete dominance relations or can easily be extended to them. In this paper, we deliver relation-algebraic specifications of the following choice sets: Condorcet non-losers, the Schwartz set, the top cycle, the uncovered set, the minimal covering set, the Banks set, and the tournament equilibrium set. Moreover, the above choice sets are visualized via the RelView tool for a tournament on a small set of alternatives to give an impression of the possibilities and features of RelView in this regard. Even in this small example computing by hand the choice sets just mentioned would already be a major task with a high risk of making mistakes. RelView guarantees us correct solutions, because it directly uses the mathematical relation-algebraic equations, which have been proved to be correct by formal calculations.

The remainder of the paper is organized as follows. First, the essential relation-algebraic preliminaries are introduced in Section 2. This includes the relation-algebraic modeling of sets and Cartesian products and the introduction of some notions that will be useful for the problems we want to solve. In Section 3 we first describe some well-known concepts for tournament winners. To give an impression of RelView's visualization potential with respect to the computation of choice sets, we then show a series of pictures produced by the tool. The corresponding relation-algebraic expressions are presented in Section 4. We demonstrate how to calculate them from formal logical problem specifications and how to translate them into the programming language of the tool. Section 5 sketches some generalizations and contains some concluding remarks.

2. Relation-algebraic preliminaries

In this section we provide the relation-algebraic preliminaries as used throughout this paper. In particular, we focus on the modeling of sets and Cartesian products which are not commonly used and hence require some detailed explanation. More details can be found, for example, in [30,31].

2.1. Fundamentals of binary relations

We write $R: X \leftrightarrow Y$ if R is a (typed and binary) relation with source X and target Y, i.e., a subset of the Cartesian product $X \times Y$, and $[X \leftrightarrow Y]$ for the *type* of all these relations, i.e., the powerset $2^{X \times Y}$. We may consider R also as a Boolean $|X| \times |Y|$ matrix if its carrier sets X and Y are finite. This interpretation is well suited for many purposes and Boolean matrices are also used as one of the graphical representations of relations within RelView. Therefore, in this paper we often use Boolean matrix terminology and notation. In particular, we speak of rows, columns and entries of relations and write $R_{x,y}$ instead of $\langle x, y\rangle \in R$ or xRy. To avoid pathologic cases, in the following we assume that all carrier sets of relations are non-empty.

We will use the following basic operations on relations (cf. [30,31]): $\overline{R}$ (*complement*, *negation*), $R \cup S$ (*union*, *join*), $R \cap S$ (*intersection*, *meet*), R^{T} (*transposition*, *converse relation*) and $R; S$ (*composition*, *multiplication*). Furthermore, we will use the special relations O (*empty relation*), L (*universal relation*), and I (*identity relation*). Here we overload the symbols, i.e., we avoid the binding of types to them. Finally, if $R: X \leftrightarrow Y$ is included in $S: X \leftrightarrow Y$ we write $R \subseteq S$ and equality of R and S is denoted as $R = S$. In order to reduce the use of brackets, it is generally agreed that composition binds stronger than union and intersection. So, for instance, $R \cup S; T$ stands for $R \cup (S; T)$ and not for $(R \cup S); T$. Similarly, $R; S \cap T$ should be read as $(R; S) \cap T$ and not as $R; (S \cap T)$.

A relation $R: X \leftrightarrow X$ is *asymmetric* if $R \subseteq \overline{R^{\mathsf{T}}}$, *irreflexive* if $R \subseteq \overline{\mathsf{I}}$, *transitive* if $R; R \subseteq R$ and *complete* if $\overline{\mathsf{I}} \subseteq R \cup R^{\mathsf{T}}$. These are the relation-algebraic (or point-free) specifications of well-known properties which usually are defined point-wisely. For instance, $\overline{\mathsf{I}} \subseteq R \cup R^{\mathsf{T}}$ specifies that for all $x, y \in X$, from $x \neq y$ it follows $R_{x,y}$ or $R_{y,x}$, i.e., different elements are related via R. The asymmetry of R implies its irreflexivity, and in this case the completeness of R is equivalent to $\overline{\mathsf{I}} = R \cup R^{\mathsf{T}}$.

Finally, we need the *transitive closure* $R^{+}: X \leftrightarrow X$ of a relation $R: X \leftrightarrow X$. This is the least (with respect to inclusion) transitive relation of type $[X \leftrightarrow X]$ that contains R. Via the powers of R, inductively defined by $R^0 := \mathsf{I}$ and $R^{i+1} := R; R^i$ for all $i \in \mathbb{N}$, we can specify R^{+} by $R^{+} = \bigcup_{i>0} R^i$.

2.2. Modeling sets

Relation algebra offers different ways of modeling sets and subsets of sets. Our first modeling uses so-called *vectors*, which are relations v with $v = v; \mathsf{L}$. Since for a vector the range is irrelevant, we consider in the following mostly vectors $v: X \leftrightarrow \mathbf{1}$ with a specific singleton set $\mathbf{1} = \{\perp\}$ as target and omit in such cases the subscript $\perp$, i.e., we write v_x instead of $v_{x,\perp}$. Such a vector can be considered as a Boolean matrix with exactly one column, i.e., as a Boolean column vector, and *represents* the subset $\{x \in X | v_x\}$ of X. A non-empty vector v is a *point* if $v; v^{\mathsf{T}} \subseteq \mathsf{I}$, i.e., if it is *injective*. This means that it represents a singleton subset of its source or an element from it if we identify a singleton set $\{x\}$ with the element x. In the Boolean matrix model, hence, a point $v: X \leftrightarrow \mathbf{1}$ is a Boolean column vector in which exactly one entry is 1.

As a second way to model sets, we will apply the relation-level equivalents of the set-theoretic symbol $\in$, that is, *membership-relations* $\mathsf{M}: X \leftrightarrow 2^X$. These specific relations are defined by demanding for all $x \in X$ and $Y \in 2^X$ that $\mathsf{M}_{x,Y}$ iff $x \in Y$. A simple Boolean matrix implementation of membership-relations requires an exponential number of bits. However, in [25] an implementation of $\mathsf{M}: X \leftrightarrow 2^X$ using BDDs is presented, where the number of BDD-vertices is linear in the size of the base set X. This implementation is part of the RelView tool.

Finally, we will use embeddings for modeling sets. Given an injective function $\imath: Y \to X$ (in the usual mathematical sense), we may consider Y as a subset of X by identifying it with its image under $\imath$. If Y is actually a subset of X and $\imath$ is given as a relation of type $[Y \leftrightarrow X]$ such that $\imath_{y,x}$ iff $y = x$ for all $y \in Y$ and $x \in X$, then the vector $\imath^{\mathsf{T}}; \mathsf{L}: X \leftrightarrow \mathbf{1}$ represents Y as a subset of X in the sense above. Clearly, the transition in the other direction is also possible, i.e., the generation of an *embedding-relation* $inj(v): Y \leftrightarrow X$ from the vector representation $v: X \leftrightarrow \mathbf{1}$ of the subset Y of X such that for all $y \in Y$ and $x \in X$ we have $inj(v)_{y,x}$ iff $y = x$. We only have to remove from the identity relation $\mathsf{I}: X \leftrightarrow X$ all x-rows where the element x ranges over the set $X \backslash Y$.

Considered as a single vector of type $[X \leftrightarrow \mathbf{1}]$, each column of $\mathsf{M}: X \leftrightarrow 2^X$ represents a single subset of X, so that M *column-wise represents* the powerset 2^X. A combination of embedding-relations with membership-relations allows to extend this to a column-wise representation of subsets of powersets. More specifically, if the vector $v: 2^X \leftrightarrow \mathbf{1}$ represents a subset $\mathfrak{S}$ of 2^X in the sense above and we define the relation $S: X \leftrightarrow \mathfrak{S}$ by $S := \mathsf{M}; inj(v)^{\mathsf{T}}$, then for all $x \in X$ and $Y \in \mathfrak{S}$ we have $S_{x,Y}$ iff $x \in Y$. This means that the elements of $\mathfrak{S}$ are represented precisely by the columns of S. A further consequence is that $\overline{S^{\mathsf{T}}; \overline{S}}: \mathfrak{S} \leftrightarrow \mathfrak{S}$ is the relation-algebraic