Journal of Logical and Algebraic Methods in Programming 90 (2017) 102–124

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Using relation-algebraic means and tool support for investigating and computing bipartitions

Rudolf Berghammer [a,*], Insa Stucke [a], Michael Winter [b]

[a] Institut für Informatik, Christian-Albrechts-Universität zu Kiel, Olshausenstraße 40, 24098 Kiel, Germany
[b] Department of Computer Science, Brock University, St. Catharines, ON, Canada

ARTICLE INFO

Article history:
Received 1 February 2016
Received in revised form 27 February 2017
Accepted 18 April 2017
Available online 10 May 2017

Keywords:
Dedekind category
Relation algebra
Bipartition
Relational axiom of choice
Splitting
Mechanised proofs

ABSTRACT

Using Dedekind categories as an algebraic structure for (binary) set-theoretic relations without complements, we present purely algebraic definitions of "to be bipartite" and "to possess no odd cycles" and prove that both notions coincide in case that the reflexive–transitive closure of the relation in question is symmetric. This generalises D. Kőnig's well-known theorem from undirected graphs to abstract relations and, hence, to models such as *L*-relations that are different from set-theoretic relations. One direction of this generalisation holds for general relations. The other direction requires the symmetry of the reflexive–transitive closure as premise and is shown by specifying a bipartition for the given relation in form of a pair of disjoint relational vectors. For set-theoretic relations this immediately leads to a relational program for computing a bipartition. We also investigate the structure of a certain set of bipartitions that is generated by a given bipartition, explore the set of all bipartitions of a bipartite relation with respect to minimality and maximality of elements in view of the component-wise inclusion, show a relationship between bipartitions and bichromatic partitions and also discuss how the algebraic proofs can be mechanised using theorem proving tools.

1. Introduction

Based on pioneering work of mainly G. Boole, A. de Morgan, C. Peirce and E. Schröder in the 19th century, the modern axiomatic investigation of the calculus of (binary) relations started with the seminal paper [31] of A. Tarski in the middle of the 20th century. Since many years this calculus is widely used by mathematicians, computer scientists and engineers as a conceptual and methodological base for investigating fundamental notions and problem solving. A lot of examples and references to relevant literature can be found, for instance, in the textbooks [3,29,30] and the proceedings of the international conferences RAMiCS.

Relation algebra, the axiomatic algebraic structure underlying the calculus of relations, has been applied to many concrete examples, particularly to graph-theoretic problems. This is mainly due to the fact that a directed graph can be seen as a binary relation on the vertex set. Other kinds of graphs, like undirected graphs, hypergraphs and multigraphs, and specific classes of graphs, like trees, forests and bipartite graphs, can also be modelled easily using relation algebra as, e.g., demonstrated in [3,29,30]. These investigations have been accompanied by tool support. The latter concerns the mechanisation of relation algebra and the execution of relational programs in tools like the Kiel RelView system (see [2,20,38]) for

* Corresponding author.
E-mail address: rub@informatik.uni-kiel.de (R. Berghammer).

http://dx.doi.org/10.1016/j.jlamp.2017.04.001
2352-2208/

set-theoretic relations, i.e., sets of pairs, and the generic matrix manipulator developed at Brock University (see [16,17]) for dealing with more general structures than classical relation algebra. (In later sections by a relational program we always mean a while-program with relations as the main datatype, i.e., a program that immediately can be translated into RelView-code). Tool support also concerns mechanised theorem proving in the context of relation algebra and more general structures. See e.g., [5–7,12,15] for some applications. The use of automated theorem provers or proof assistant tools in the context of such algebraic structures is mainly based on the fact that (relation-)algebraic proofs usually are very formal and precise and calculational transformations frequently constitute their decisive parts.

In this paper, which is an extended version of [8], we continue this line of research. Primarily, we prove some results concerning the bipartition of relations – which are regarded as abstractions of directed graphs – with purely relation-algebraic means, that is, without any reference to the fact that relations are sets of pairs over certain carrier sets. That relations are sets of pairs and specify directed graphs is only used for explanatory purposes and for describing certain algebraic constructions. In such constructions, the algebraic calculations and the proofs we even avoid the use of complements of relations which, algebraically, means that we do not work with relation algebra in the sense of [31,32] (homogeneous approach) or [29,30] (heterogeneous approach), but with the more general algebraic structure of a Dedekind category. This algebraic structure has been introduced in [23] and is, for example, used in [14] to investigate crispness of L-relations and in [34] to model processes. L-relations generalise fuzzy relations by replacing the unit interval $[0, 1]$ of the real numbers as the domain of membership by an arbitrary complete distributive lattice $(L, \vee, \wedge, 0, 1)$. In the well-known matrix model of relations fuzzy relations are matrices with entries from the unit interval $[0, 1]$ of $\mathbb{R}$, whereas L-relations are matrices with entries from a suitable lattice L. For L being the two-element Boolean lattice of truth values, an L-relation can be seen as a set-theoretic relation. Since L-relations form a Dedekind category, our results also apply to this generalisation of set-theoretic relations.

Considering abstract relations as morphisms of a Dedekind category, we present purely algebraic definitions of the notions "to be bipartite" and "to possess no odd cycles". For set-theoretic relations the second notion coincides with the corresponding notion from graph theory. Since we do not use complements, however, our notions of a bipartition of a relation and of "to be bipartite" are a bit more general than the graph-theoretic ones. In terms of graphs they are referring to non-isolated vertices only. This approach implies that each bipartition of a given relation R generates certain new bipartitions of R, which then can be ordered in a natural fashion. We not only investigate the structure of these generated sets of bipartitions but also explore the set of all bipartitions of R with respect to minimality and maximality of elements in view of the above mentioned order. All that is done in Section 3 after the introduction of the preliminaries from relation algebras and Dedekind categories in Section 2.

As a main result of the paper, in Section 4 we algebraically prove for all relations with a symmetric reflexive–transitive closure, i.e., in particular for all symmetric relations, that they are bipartite if and only if they do not possess odd cycles. This generalises D. Kőnig's well-known theorem (published in [18]) from undirected graphs to Dedekind categories. One direction of this generalisation holds for general relations. The other direction requires the symmetry of the reflexive–transitive closure as premise and is shown by specifying a bipartition for the given relation in form of a pair of disjoint relational vectors. This is done by means of algebraic expressions. When using the RelView system, in case of symmetric set-theoretic relations (i.e., undirected graphs) these are based on the algebraic construction of a splitting, that generalises projections of set-theoretic equivalence relations and in RelView is not available as a pre-defined operation. But splittings easily can be computed by means of a simple relational program.

In Section 5 we present an example for the results of Section 3 and Section 4 using L-relations. In it an L-relation specifies a relationship between two elements – here persons – that has two aspects. Concerning Section 3, this example describes a generated ordered set of bipartitions, concerning Section 4 it demonstrates the two decisive theorems of this section but also shows a situation in which two other decisive results of it cannot be applied.

After that, in Section 6 we take the above mentioned simple relational program for computing splittings and derive from it a relational program that computes for a set-theoretic relation with a symmetric reflexive–transitive closure and without odd cycles a relational vector that, together with its complement, constitutes a bipartition of the input. The program immediately can be translated into the programming language of RelView such that the system can be used for computing bipartitions.

Following [1], in Section 7 we consider bichromatic graphs as generalisations of bipartite graphs and define the notions of a bichromatic relation and a bichromatic partition of a relation with algebraic means. Then we show how the test of "to be bichromatic" and the computation of a bichromatic partition can be reduced to the test of "to be bipartite" and the computation of a bipartition. All that is again done in a purely algebraic way.

We already have mentioned that algebraic calculations concerning relations are very formal and precise and calculational transformations frequently constitute their decisive parts. This not only minimises the danger of making errors within proofs, but also allows, as we also have mentioned above, the use of theorem provers and proof assistant tools. In [8] we have used the automated theorem prover Prover9 (see [37]) for first-order classical logic and the proof assistant tool Coq (see [9,36]) for a derivative of the calculus of constructions to check our results and we also shortly have reported on our experience in respect thereof. In Section 8 of the present paper we describe the use of these two tools and our experience with them in more detail and do the same also for a third tool, the first-order automated theorem prover Vampire (see [27,39]).