Cardinalities of Finite Relations in Coq with Applications

Paul Brunet, Damien Pous, Insa Stucke

Cardinalities of Finite Relations in Coq with Applications *

Paul Brunet
Univ. Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, UMR 5668

Damien Pous
Univ. Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, UMR 5668

Insa Stucke
Institut für Informatik, Christian-Albrechts-Universität zu Kiel, Germany

Abstract

In this paper we present an extension of a Coq library for relation algebras and related algebraic structures. So far, the library did not provide any tools about the cardinalities of relations. Thus we add an algebraic axiomatisation of cardinalities. Its point-free nature makes it possible to reason about cardinal purely algebraically, which is well-suited for mechanisation. We present several applications, in the area of graph theory and program verification.

1 Introduction

Binary relations have a rich algebraic structure: rather than considering relations as objects relating points, one can see them as abstract objects that can be combined using various operations (e.g., union, intersection, composition, transpose). Those operations are subject to many laws (e.g., associativity, distributivity). One can thus use equational reasoning to prove results about binary relations, graphs, or programs manipulating such structures. This is the so-called relation-algebraic method [25–27]. It has been successfully applied in social choice [11], game theory [1], and functional programming [3, 10].

An important advantage of such an approach is that it is more amenable to automation and mechanisation than the standard approach using points and first-order logic, see, for instance, [6, 17]. In, for example, [5, 7] it turned out that such an approach is

*An extended abstract of this article is published in the proceedings of the 7th International Conference on Interactive Theorem Proving (ITP 2016), see [9]. The first two authors were supported by the European Research Council (ERC) under the Horizon 2020 programme (CoVeCe, grant agreement No 678157) and the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007)

very limited, in particular when it comes to heterogeneous relations and to more complex proofs. Due to this a change to systems providing typed relations becomes necessary.

Lately, the second author developed a library for the Coq proof assistant [21, 22], allowing one to develop large proofs using the relation algebraic approach. This library contains powerful automation tactics for some decidable fragments of relation algebra (Kleene algebra and Kleene algebra with tests), normalisation tactics, and tools for rewriting modulo associativity of relational composition.

The third author recently relied on this library to formalise algebraic correctness proofs for several standard algorithms from graph theory. For instance, in [4] a program for computing vertex colourings is considered by using relation-algebraic reasoning. Furthermore, in [7] there can be found an algebraic investigation of computing bipartitions with the help of Dedekind categories.

In the present paper, we show how to extend this library to deal with cardinals of relations, thus allowing one to reason about quantitative aspects. To illustrate the benefits of such an approach, we first show basic results about the sizes of linear orders and trees. Then we study the notion of independent sets from graph theory: we prove the correctness of a standard algorithm for finding them, and we establish standard bounds about their sizes. Last, we prove a decomposition theorem for relations.

Outline. We start by recalling the basic definitions about binary relations, and the axiomatisation of relation algebra we use in the sequel (Section 2). Then we describe how this axiomatisation is formalised in Coq, and how we setup various tools that ease subsequent developments (Section 3). We present our case-studies in Section 4, and we conclude in Section 5.

The Coq library associated to this paper is available online:

`http://media.informatik.uni-kiel.de/cardinal/`

2 Preliminaries

Given two sets X, Y, a *binary relation* is a subset $R \in \mathcal{P}(X \times Y)$. The set X (resp. Y) is called the *domain* (resp. *codomain*) of the relation.

With the usual set-theoretic operations of inclusion ($\subseteq$), union ($\cup$), intersection ($\cap$), complement ($\bar{\cdot}$), the empty relation (O_{XY}) and the universal relation (L_{XY}), the binary relations between two sets X and Y form a Boolean lattice.

Given three sets X, Y, Z, the *composition* of two relations $R \in \mathcal{P}(X \times Y)$ and $S \in \mathcal{P}(Y \times Z)$ is defined as follows:

$$RS \triangleq \{(x, z) \mid \exists\, y \in Y, (x, y) \in R \wedge (y, z) \in S\} \in \mathcal{P}(X \times Z)$$

Accordingly, given a set X, the *identity relation on* X is the diagonal relation

$$\mathsf{I}_X \triangleq \{(x, x) \mid x \in X\} \in \mathcal{P}(X \times X)$$

If they can be inferred from the context we omit type annotations for the symbols denoting constants, thus writing L, O, or I.

Transposition exchanges domain and codomain: for $R \in \mathcal{P}(X \times Y)$,

$$R^{\mathsf{T}} \triangleq \{(y, x) \mid (x, y) \in R\} \in \mathcal{P}(Y \times X)$$

Finally, *reflexive transitive closure* is an operation on *homogeneous* relations, those relations with the same domain and codomain: for $R \in \mathcal{P}(X \times X)$,

$$R^{\star} \triangleq \{(x_0, x_n) \mid \exists\, x_1, \ldots, x_{n-1},\ \forall\, i \in \mathbb{N}_{<n},\ (x_i, x_{i+1}) \in R\} \in \mathcal{P}(X \times X)$$

These operations can be abstracted through the axiomatic notion of *relation algebra*. Binary relations being the standard model of such an algebra, we use the same notations.

Definition 2.1 (Relation Algebra)**.** *A* relation algebra (with Kleene star) *is a category whose homsets are Boolean lattices, together with two operations of* transposition *$(\cdot^{\mathsf{T}})$ and* Kleene star *$(\cdot^{\star})$ such that:*

($P1$) *composition is monotone in its two arguments, distributes over unions and is absorbed by the bottom elements;*

($P2$) *transposition maps every morphism $R : X \to Y$ into a morphism $R^{\mathsf{T}} : Y \to X$, is monotone, involutive ($R^{\mathsf{T}\mathsf{T}} = R$), and reverses compositions: for all morphisms R, S of appropriate types, we have $(RS)^{\mathsf{T}} = S^{\mathsf{T}} R^{\mathsf{T}}$;*

($P3$) *Kleene star maps every homogeneous morphism $R : X \to X$ into a morphism $R^{\star} : X \to X$ such that $\mathsf{I} \cup R R^{\star} \subseteq R^{\star}$ and for all object Y and morphism $S : X \to Y$, $RS \subseteq S$ entails $R^{\star} S \subseteq S$;*

($P4$) *for all morphisms Q, R, S of appropriate types,*

$$QR \subseteq S \qquad \textit{iff} \qquad Q^{\mathsf{T}} \overline{S} \subseteq \overline{R} \qquad \textit{iff} \qquad \overline{S}\, R^{\mathsf{T}} \subseteq \overline{Q}\,;$$

($P5$) *for every morphism $R : X \to Y$, $R \neq \mathsf{O}$ if and only if for all objects X', Y', $\mathsf{L} R \mathsf{L} = \mathsf{L}_{X'Y'}$.*

From properties ($P2$), we deduce that transposition commutes with all Boolean connectives, and that $\mathsf{I}^{\mathsf{T}} = \mathsf{I}$.

Properties ($P3$) characterise Kleene star. In fact, together with conditions ($P2$), they entail their symmetrical counterparts ($\mathsf{I} \cup R^{\star} R \subseteq R^{\star}$, and $SR \subseteq S$ entails $SR^{\star} \subseteq S$), so that we recover standard Kleene algebra axioms [20].

Equivalences ($P4$) are called *Schröder equivalences* in [25]; they correspond to the fact that the structure is residuated [14]. The last property ($P5$) is known as *Tarski's rule*; it makes it possible to reason algebraically about non-emptiness.

Important classes of morphisms can be defined algebraically. In the sequel, we use the following terminology: a homogeneous morphism $R : X \to X$ is

- *reflexive* if $\mathsf{I} \subseteq R$,
- *symmetric* if $R^{\mathsf{T}} \subseteq R$,

- *transitive* if $RR \subseteq R$.

One can easily check that these definitions correspond to the standard definitions in the model of binary relations. For instance, for reflexivity we have

$$\begin{aligned}
&R \in \mathcal{P}(X \times X) \text{ is reflexive} \\
&\iff \forall\, x \in X,\ (x, x) \in R \\
&\iff \forall\, x, y \in X,\ x = y \Rightarrow (x, y) \in R \\
&\iff \forall\, x, y \in X,\ (x, y) \in \mathsf{I} \Rightarrow (x, y) \in R \\
&\iff \mathsf{I} \subseteq R
\end{aligned}$$

We shall see other classes of homogeneous morphisms in Section 4. We also say that an arbitrary morphism $R : X \to Y$ is:

- *injective* if $RR^{\mathsf{T}} \subseteq \mathsf{I}$,
- *surjective* if $\mathsf{I} \subseteq R^{\mathsf{T}}R$,
- *univalent* if its transpose is injective (i.e., $R^{\mathsf{T}}R \subseteq \mathsf{I}$),
- *total* if its transpose is surjective (i.e., $\mathsf{I} \subseteq RR^{\mathsf{T}}$),
- a *mapping* if R is total and univalent.

Again, these definitions correspond to the standard definitions in the model of binary relations. For instance, for injectivity we have

$$\begin{aligned}
&R \in \mathcal{P}(X \times Y) \text{ is injective} \\
&\iff \forall x, y \in X,\ \forall z \in Y,\ (x, z) \in R \wedge (y, z) \in R \Rightarrow x = y \\
&\iff \forall x, y \in X,\ (\exists z \in Y,\ (x, z) \in R \wedge (y, z) \in R) \Rightarrow x = y \\
&\iff \forall x, y \in X,\ (\exists z \in Y,\ (x, z) \in R \wedge (z, y) \in R^{\mathsf{T}}) \Rightarrow x = y \\
&\iff \forall x, y \in X,\ (x, y) \in RR^{\mathsf{T}} \Rightarrow (x, y) \in \mathsf{I} \\
&\iff RR^{\mathsf{T}} \subseteq \mathsf{I}
\end{aligned}$$

Before introducing cardinals, we need a way to abstract over the singleton sets from the model of binary relations. Therefor we use the following definition:

Definition 2.2 (Unit in a Relation Algebra)**.** *A* unit *in a relation algebra is an object* 1 *such that* $\mathsf{O}_{11} \neq \mathsf{L}_{11}$ *and* $\mathsf{I}_1 = \mathsf{L}_{11}$.

In other words, there are only two morphisms from a unit to itself. In the model of binary relations, every singleton set is a unit.

Using units, we can axiomatise the notion of cardinal in a relation algebra. To this end, we mainly follow Kawahara [19]:

Definition 2.3 (Cardinal)**.** *A relation algebra with* cardinal *is a relation algebra with a unit* 1 *and a monotone function* $|\cdot|$ *from morphisms to natural numbers such that for all morphisms* Q, R, S *of appropriate types:*

$(C1)$ $|\mathsf{O}| = 0$,

$(C2)$ $|\mathsf{I}_1| = 1$,

$(C3)$ $|R^\mathsf{T}| = |R|$,

$(C4)$ $|R \cup S| + |R \cap S| = |R| + |S|$,

$(C5)$ *if* Q *is univalent, then* $|R \cap Q^\mathsf{T}S| \leq |QR \cap S|$ *and* $|Q \cap SR^\mathsf{T}| \leq |QR \cap S|$.

A difference with Kawahara's axioms is that we do not require $|R| = 0$ to entail $R = \mathsf{O}$, as this follows from the assumptions of Definition 2.4 below. Note that these requirements for a cardinal rule out infinite binary relations: we have to restrict to binary relations between finite sets. Typically, in this model of (finite) binary relations, the cardinal of a relation is the number of pairs it contains. This restriction is harmless in practice: we only work with finite sets when we study, for example, algorithms.

Many natural facts of cardinal can be derived just from conditions $(C1)$ to $(C4)$. For example we get $|\bigcup_{R\in\mathcal{R}} R| = \sum_{R\in\mathcal{R}} |R|$ for every finite set $\mathcal{R}$ of pairwise disjoint morphisms. The last condition $(C5)$ is less intuitive; it is called the *Dedekind inequality* in [19]. It allows one to compare cardinalities of morphisms of different types. Kawahara uses it to obtain, for instance, the following result, which we need in the sequel:

Lemma 2.1. *Assume a relation algebra with cardinal. For all morphisms* Q, R, S *of appropriate type, we have:*

1. *If* R *and* S *are univalent, then* $|RS \cap Q| = |R \cap QS^\mathsf{T}|$.
2. *If* R *is univalent and* S *is a mapping, then* $|RS| = |R|$.

Leaving cardinals aside, two important classes of morphisms are that of vectors and points, as introduced in [24], for providing a way to model subsets and single elements of sets, respectively:

- *vectors*, denoted with lower case letters v, w in the sequel, are morphisms $v : X \to Y$ such that $v = v\mathsf{L}$. In the standard model, this condition precisely amounts to being of the special shape $V \times Y$ for a subset $V \subseteq X$.
- *points*, denoted with lower case letters p, q in the sequel, are injective and nonempty vectors. In the standard model, this condition precisely amounts to being of the special shape $\{x\} \times Y$ for an element $x \in X$.

Hence in the binary relations model there is a one-on-one correspondence between vectors from X to Y and subsets of X, points being the vectors associated with singleton sets. Another characterisation can be described from the Boolean-matrix representation of binary relations: a vector is a matrix whose rows are either zero everywhere or one everywhere, and a point is a matrix with a single row of ones and zeros everywhere else.

Also note that from Tarski's rule $(P5)$ and non-emptiness, one deduces that points are surjective, thus the transpose of a point is a mapping.

Every morphism with unit as its codomain is a vector. As expected, points with unit as their codomain have cardinal one:

Lemma 2.2. *Let $p : X \to 1$ be a point in a relation algebra with a cardinal (and unit). We have $|p| = 1$.*

Proof. We have $|p| = |p^\mathsf{T}| = |\mathsf{I}_1 p^\mathsf{T}| = |\mathsf{I}_1| = 1$, using cardinality axiom (C3), Lemma 2.1(2) (I_1 is univalent and $p^\mathsf{T} : 1 \to X$ is a mapping) and cardinality axiom (C2). □

We conclude this preliminary section with the notion of pointed relation algebra. Indeed, in the model of binary relations, the universal relation between X and Y is the least upper bound of all points between X and Y. This property is called the *point axiom* in [13]. Since we restrict to finite relations, we give a finitary presentation of this law.

Definition 2.4 (Pointed Relation Algebra)**.** *A relation algebra is* pointed *if for all X, Y there exists a (finite) set P_{XY} of points such that*

$$\mathsf{L}_{XY} = \bigcup_{p \in P_{XY}} p \ .$$

A typical result in pointed relation algebras is that the identities decompose as follows:

$$\mathsf{I}_X = \bigcup_{p \in P_{XX}} pp^\mathsf{T}$$

When working in pointed relation algebras with cardinal, we also have results like the following, where we use $|X|$ as a shorthand notation for $|\mathsf{L}_{X1}|$:

Lemma 2.3.

1. *For all objects X and Y we have $|\mathsf{L}_{XY}| = |X| \cdot |Y|$.*
2. *For every object X we have $|\mathsf{I}_X| = |X|$.*

In fact pointed relation algebras always have a cardinal:

Proposition 2.1. *Avery pointed relation algebra with unit has a (unique) cardinal given by the function*

$$(R : X \to Y) \ \mapsto \ \#\left\{(p, q) \mid p \in P_{X1},\ q \in P_{Y1},\ pq^\mathsf{T} \subseteq R\right\} \ .$$

3 Relation Algebra in Coq

The Coq library RelationAlgebra [21, 22] provides axiomatisations and tools for various fragments of the calculus of relations: from ordered monoids to Kleene algebra, residuated structures, and Dedekind Categories. It is structured in a modular way: one can easily decide which operations and axioms to include.

In the present case, these are Boolean operations, composition, identities, transpose, and Kleene star. We extended the library with a module `relalg` containing definitions and facts about this particular fragment. For instance, this module defines many classes of relations, some of which we already mentioned in Section 2. For those properties we use classes in Coq:

```
Class is_vector (C: ops) X Y (v: C X Y) := vector: v*top == v.
```

Here we assume an ambient relation algebra `C`, `ops` being the corresponding notion, as exported by the RelationAlgebra library. Variables `X`,`Y` are objects of the category, and `v: C X Y` is a morphism from `X` to `Y`. The symbols `*` and `==` respectively denote composition and equality; `top` is the top morphism of appropriate type: its source and target (`Y` twice) are inferred automatically.

We use a class to ease rewriting in subsequent proofs: in this case, the tactic `rewrite vector` will look for a subterm of a shape `v*top` with `v` provably a vector using typeclass resolution, and replace it with `v`. Similar classes are set-up for all notions discussed in the sequel (injective, surjective, univalent, total, mapping, points, and much more).

The library does not provide Tarski's rule $(P5)$, which is not positive. Still, many results can be obtained by using the following definition of non-emptiness:

```
Class is_nonempty (C: ops) X Y (R: C X Y) :=
  nonempty: forall p q, top' p q <== top * R * top.
```

(In the second line, the notation `top' p q` is used to specify the source and the target of the top morphism: they cannot be inferred automatically. The symbol `<==` denotes the preorder $\subseteq$.) Some proofs however require us to link this algebraic definition of non-emptiness to the ambient logic, using Tarski's rule. For this purpose we use the following class:

```
Class tarski (C: ops) :=
  Tarski: forall X Y (R: C X Y), ~(R <== 0) ↔ is_nonempty R.
```

(Here, the prefix symbol ~ is Coq's negation—thus in the ambient logic.)

The RelationAlgebra library provides several automation tactics to ease equational reasoning [21, 22]. The most important ones are:

- `ra_normalise` for normalising the current goal w.r.t. the simplest laws (mostly about idempotent semirings, units and transposition),
- `ra` for solving goals by normalisation and comparison,
- `lattice` for solving lattice-theoretic goals,
- `mrewrite` for rewriting modulo associativity of categorical composition,
- `ka` for solving goals about Kleene algebra, using automata algorithms.

For an example of the latter tactic, consider the law $R^\star = (RR)^\star \cup R(RR)^\star$. This equation actually belongs to the equational theory of Kleene algebra and can be proved automatically by `ka`.

```
Lemma decomp_str (C: ops) X (R: C X X): R^* == (R*R)^* ∪ R*(R*R)^*.
Proof. ka. Qed.
```

While giving an explicit proof from axioms $(P1)$ and $(P3)$ is not overwhelming, it already requires a few lines and some non-trivial reasoning. In larger proofs, it is thus extremely beneficial to be able to rely on such an automated tactic.

We define three more classes to represent relation algebra with unit, relations algebra with cardinal, and pointed relation algebra. Units are introduced as follows:

```
Class united (C: ops) := {
 unit: ob C;
 top_unit: top' unit unit == 1;
 nonempty_unit:> is_nonempty (top' unit unit) }.
```

The field `unit` is the unit object; the two subsequent fields correspond to the requirements from Definition 2.2. The symbol `1` is our notation for identity morphisms.

Assuming units, one can then define cardinals:

```
Class cardinal (C: ops) (U: united C) := {
 card: forall X Y, C X Y → nat;
 card0: forall X Y, @card X Y 0 = 0;
 card1: @card unit unit 1 = 1;
 cardcnv: forall X Y (R: C X Y), card R° = card R;
 cardcup: forall X Y (R S: C X Y),
     card (R ∪ S) + card (R ∩ S) = card R + card S;
 cardded: forall X Y Z (R: C X Y) (S: C Y Z) (T: C X Z),
     is_injective R → card (T ∩ (R⋅S)) ≤ card (R° ⋅ T ∩ S);
 cardded': forall X Y Z (R: C Y X) (S: C Y Z) (T: C Z X),
     is_univalent R → card (R ∩ (S⋅T)) ≤ card (R ⋅ T° ∩ S) }.
```

The first field is the cardinal operation itself. The remaining ones correspond to the conditions from Definition 2.3. (Dedekind inequalities $(C5)$ are split into two fields and typeset slightly differently to ease rewriting.)

Next we give three Coq proofs about cardinals, to illustrate the ease with which it is possible to reason about them. The first two correspond to Lemma 2.1.

```
Lemma card_uniuni X Y Z (R: C X Y) (S: C Y Z) (T: C X Z):
      is_univalent R → is_univalent S → card (R⋅(S ∩ T)) = card (R ∩ (T⋅S°)).
Proof.
  apply antisym.
   rewrite ←cardcnv, cnvcap, cnvdot, capC.
   rewrite cardded, ←cardcnv. apply card_leq. ra.
  rewrite cardded'. apply card_leq. ra.
Qed.
```

The lemma `antisym` makes it possible to prove an equality by double inclusion; `cnvcap` and `cnvdot` state that transpose commutes with meets and composition; `capC` states commutativity of meet; `card_leq` correlates with monotonicity of the cardinal operation.

```
Lemma card_unimap X Y Z (R: C X Y) (S: C Y Z):
      is_univalent R → is_mapping S → card (R⋅S) = card R.
Proof.
  rewrite ←capxt, card_uniuni, surjective_tx. apply card_weq. ra.
Qed.
```

Lemma `capxt` states that top is a unit for meet; `surjective_tx` that every surjective morphism R satisfies $\mathsf{L}R = \mathsf{L}$; and `card_weq` that cardinals are preserved by equality.

The third Coq proof we show here is that of Lemma 2.2. It follows precisely the one we gave earlier.

```
Lemma card_point X (R: C X unit): is_point R → card R = 1.
```

```
Proof.
  rewrite ←cardcnv, ←dot1x. rewrite card_unimap. apply card1.
Qed.
```

Lemma `dot1x` states that `I` is a left unit for composition.

Finally, pointed relation algebras are defined by requiring a finite set of points for each homset, of which `L` is the least upper bound. We implement such sets by duplicate-free lists: this may not come as a surprise to literate Coq users, as lists are integrated into the Coq system, and thus quite convenient to use.

```
Class pointed (C: ops) := {
 points: forall X Y, list (C X Y);
 points_points: forall X Y p, In p (points X Y) → is_point p;
 points_nodup: forall X Y, nodup (points X Y);
 point_ax: forall X Y, top == \sup_(p\in points X Y) p }.
```

To manipulate this (finite but not binary) least upper bound, we rely on the support already provided in RelationAlgebra, largely inspired from "big operators" in mathcomp/ssreflect [8, 15]. Whence the `\sup_(...)...` notation in the field `point_ax`.

4 Applications

In this section we present some applications of the usage of cardinals. We start with an easy example where we link the cardinality of morphisms representing linear orders or trees to the cardinality of their carrier sets. Then we present examples that are related to graph theoretical results and algorithms: the presented abstraction via relation algebra can be used in the context of program verification.

4.1 Linear orders

A morphism $R : X \to X$ is a *partial order* on X if R is *reflexive*, *antisymmetric* and *transitive* (i.e., $\mathsf{I} \subseteq R$, $R \cap R^\mathsf{T} \subseteq \mathsf{I}$ and $RR \subseteq R$). If R is additionally *linear* (i.e., $R \cup R^\mathsf{T} = \mathsf{L}$) we call R a *linear order*. Recall that for an object X, $|X|$ is a shorthand for $|\mathsf{L}_{X1}|$. We have

Theorem 4.1. *If $R : X \to X$ is a linear order, then $|R| = \frac{|X|^2+|X|}{2}$.*

Proof. Since R is antisymmetric we have $R \cap R^\mathsf{T} \subseteq \mathsf{I}$. Furthermore, we have $\mathsf{I} \subseteq R$ since R is reflexive so that $R \cap R^\mathsf{T} = \mathsf{I}$. Now we can calculate as follows:

$$\begin{aligned}
|X|^2 + |X| &= |\mathsf{L}_{XX}| + |\mathsf{I}_X| && \text{(by Lemma 2.3)}\\
&= |R \cup R^\mathsf{T}| + |\mathsf{I}_X| && (R \text{ linear})\\
&= |R \cup R^\mathsf{T}| + |R \cap R^\mathsf{T}| && (R \text{ reflexive and antisymmetric})\\
&= |R| + |R^\mathsf{T}| && \text{(by } (C4)\text{)}\\
&= |R| + |R| && \text{(by } (C3)\text{)}
\end{aligned}$$

□

With the presented tools, this lemma can be proved in Coq in a very same way. First we define a notation for the cardinal of an object:

```
Notation card' X := card (top' X unit).
```

Then we proceed as follows:

```
Lemma card_linear_order X (R: C X X): is_order R → is_linear R →
  2∗card R = card' X ∗ card' X + card' X.
Proof.
  intros Ho Hli.
  rewrite ←card_top, ←card_one.
  rewrite ←Hli.
  rewrite ←kernel_refl_antisym.
  rewrite capC, cardcup.
  rewrite cardcnv. lia.
Qed.
```

The standard Coq tactic `lia` solves linear integer arithmetic. The lemmas `card_top` and `card_one` correspond to the items of Lemma 2.3, i.e.,

```
Lemma card_top X Y: card (top' X Y) = card' X ∗ card' Y.
Lemma card_one X: card (one X) = card' X.
```

4.2 Number of edges in trees

Next, we want to show how to use relation algebra for mechanising graph theoretical results. Our first example is a well-known result about trees stating that the number of edges in a tree is exactly the number of vertices minus one.

We follow the presentation from [2]. A *forest* is an acyclic directed graph such that each vertex has at most one parent. We represent a directed graph by a morphism $R : X \to X$. Acyclicity can be expresses by using the *transitive closure* denoted with R^+ where $R^+ \triangleq R^\star R$. Then R is *acyclic* if its transitive closure is *irreflexive*, i.e., $R^+ \cap \mathsf{I} = \mathsf{O}$. Unicity of the parents is just injectivity. A *tree* is a forest with a vertex r called the *root*, such that every vertex can be reached from the root. The latter condition can be modelled algebraically by $r\mathsf{L} \subseteq R^\star$. In this setting, we want to prove

Theorem 4.2. *If $R : X \to X$ models a tree, then $|R| = |X| - 1$.*

Accordingly in Coq, we use classes to ease rewriting as before, and we define:

```
Class is_forest X (R: C X X) := {
  unique_parents:> is_injective R;
  acyclic:> is_irreflexive (R⁺) }.

Class is_tree X (R: C X X) (r: C X unit):= {
  tree_forest:> is_forest R;
  root_is_point:> is_point r;
  root_reaches_all_vertices: r∗top <== R^⋆ }.
```

We only outline the key steps in our formalisation. First, the root of a tree has no parent:

```
Lemma no_parent_to_root X (R: C X X) (r: C X unit): is_tree R r → R∗r == 0.
```

Second, the root is the only node with no parent:

```
Lemma only_root_without_parent X (R: C X X) (r p: C X unit):
  is_tree R r → is_point p → R ∗ p <== 0 → p == r.
```

Third, non-root nodes have exactly one parent:

```
Lemma card_parents_proper_node X (R: C X X) (r p: C X unit):
  is_tree R r → is_point p → p ∩ r <== 0 → card (R ∗ p) = 1.
```

The Coq proofs for these three lemmas are respectively 11, 3, and 4 lines long. We finally obtain the theorem with 16 more lines.

```
Theorem card_tree X (R: C X X) (r: C X unit): is_tree R r → card R = card' X −1.
```

In the end, the mechanised proof closely follows the paper one [2], and it is of similar size.

4.3 Approximation of maximal independent sets

The next example is motivated by the rising importance of the tool-supported verification of software. Therefore, we present a program for approximating maximum independent sets in undirected graphs. It is based on an algorithm developed and presented in [28]. In [5] a relation-algebraic formulation as well as a proof of its correctness can be found. The proof is based on the assertion-based verification method, see, e.g., [12, 16], using pre- and post-conditions, and invariants.

An *undirected (loopfree) graph* $g = (X, E)$ has a symmetric and irreflexive adjacency relation. It can thus be represented by a morphism that is symmetric ($R^\mathsf{T} \subseteq R$) and irreflexive ($R \cap \mathsf{I} = \mathsf{O}$).

An *independent set* (or *stable set*) of g is a set of vertices S such that any two vertices in S are not connected by an edge, i.e., $\{x, y\} \notin E$, for all $x, y \in S$. Furthermore, we say that an independent set S of g is *maximal* if for every independent set T of g we have $|T| \leq |S|$.

Independent sets can be modelled abstractly using vectors: a vector $s : X \to 1$ models an independent set of a morphism R if $Rs \subseteq \overline{s}$. Again, one can check the correspondence between this specification and the graph theoretical definition by equivalence transformations.

The following relation-algebraic program takes as input a relation representing an undirected graph and outputs an independent set. We assume a function *pick_point*$(\cdot)$ to pick a point $p \subseteq v$ out of a non-empty vector v, in a deterministic but unspecified manner.

Input: $R : X \to X$
1 $s := \mathsf{O}_{X1}$; $v := \mathsf{O}_{X1}$;
2 **while** $v \neq \mathsf{L}$ **do**
3 | **let** $p = \mathit{pick\_point}(\overline{v})$ **in**
4 | $s := s \cup p$; $v := v \cup p \cup Rp$;
5 **end**
6 **return** s ;

Wei's algorithm

This greedy algorithm operates by maintaining a set v (vector) of visited vertices (points), and uses a vector s to store the the independent set being built. Both these vectors are initially empty. At each step, we pick a point p in the complement of v, to ensure it has not been visited yet, and add it to s. We then add to the vector v both p and all of its neighbours. The algorithm stops when they are no more points to be picked, meaning that v encompasses all vertices of the graph.

When the maximum degree of the input graph is k, the returned independent set is at most $k+1$ times smaller than every other one. This is what we prove in the sequel.

That a (symmetric) morphism $R : X \to X$ has degree at most k is modelled by requiring that $|Rp| \leq k$ for every point $p : X \to 1$. Then we define the following predicates:

- $Inv(R, s, v)$: s is an independent set of R and $Rs \cup s = v$;
- $Post(R, s)$: s is an independent set of R and for every independent set t of R, we have $|t| \leq (k+1) \cdot |s|$.

The first predicate is the loop invariant: in addition to maintaining an independent set in s, the vector v should collects all vertices contained in the present independent set s and, additionally, their neighbours. The second predicate is the post-condition: s should be an independent set of appropriate size.

We use the standard assertion-based verification method: assuming that R is symmetric, irreflexive, and of degree at most k, we just have to show the following three proof obligations for partial correctness:

(PO1) $Inv(R, \mathsf{O}_{X1}, \mathsf{O}_{X1})$

(PO2) $Inv(R, s, v)$ and $v \neq \mathsf{L}$ imply $Inv(R, s \cup p, v \cup p \cup Rp)$, for every point $p \subseteq \overline{v}$.

(PO3) $Inv(R, s, v)$ and $v = \mathsf{L}$ imply $Post(R, s)$

For total correctness we also need to provide a loop variant to ensure termination. The cardinal of the vector v does the job: it strictly increases at each iteration:

(PO4) if p is a point satisfying $p \subseteq \overline{v}$, then $|v| < |v \cup p \cup Rp|$

Using the lemmas and tactics from RelationAlgebra as well as a small library of basic facts about cardinals, we discharge these proof obligations in Coq in a streamlined way. The proof of (PO1) is fully automatic, using the tactic `ra`; and those of (PO2), (PO3), and (PO4) are respectively 9, 9, and 6 lines long.

4.4 Independence number of a graph

The investigation of graph parameters is an important field in graph theory, for instance their dependencies and their lower and upper bounds [28]. In this section we give bounds for the *independence number* of an undirected graph $g = (X, E)$, modelled again by a symmetric and irreflexive morphism $R : X \to X$, i.e., the maximal size of an independent set defined as:

$$\alpha_R \triangleq \max \{|s| \mid s \text{ is an independent set of } R\} \quad .$$

To replay this definition in Coq, we use the support for big operations provided by the RelationAlgebra library, by declaring the (max,+) algebra on natural numbers as a model.

One easily obtain the following lower bound: $\alpha_R \leq \sqrt{|\,\overline{R}\,|}$. In fact, we have $|s| \leq \sqrt{|\,\overline{R}\,|}$ for every independent set s, which we can prove in two lines using our library.

The upper bound is harder to obtain. We have $\frac{|X|}{k+1} \leq \alpha_R$, where k is the maximum degree of R. Call *maximum* an independent set of maximal cardinal (α_R); call instead *maximal* an independent set which cannot be enlarged w.r.t. the preorder $\subseteq$:

```
Definition maximal (v: C X unit) := forall w, v <== w → R ∗ w <== !w → w <== v.
```

As expected, maximum independent sets are maximal:

```
Lemma maximum_maximal (v: C X unit):
  R∗v <== !v → card v = independent_number R → maximal v.
```

(Note that the converse is not necessarily true.) Then we prove the following algebraic characterisation of maximal independent sets: while independent sets are characterised by an inequality ($Rv \subseteq \overline{v}$), maximal are characterised by an equality ($Rv = \overline{v}$).

```
Lemma maximal_independent_iff (v: C X unit):
  R∗v <== !v → (maximal v ↔ R∗v == !v).
```

Finally, obtaining the lower bound for the independence number consists in proving that maximal independent sets, defined algebraically, satisfy this bound:

```
Lemma maximal_lower_bound (v: C X unit):
  R∗v == !v → card' X ≤ (maximum_degree R + 1) ∗ card v.
Theorem independent_lower_bound:
  card' X <== (maximum_degree R + 1) ∗ independent_number R.
```

Including the proofs of the three key lemmas, the final theorem is eventually proved in 41 lines of Coq. We consider this a success as this is comparable to what is required for a detailed paper proof.

4.5 Decomposition as a union of univalent morphisms

The last result we want to prove with the tools of Section 3 is the following:

Theorem 4.3. *For all $k \in \mathbb{N}$ and morphism $R : X \to Y$ the following facts are equivalent:*

1. *There exists a collection $\{F_1, \ldots, F_k\}$ of k pairwise disjoint and univalent morphisms with $R = \bigcup_{i=1}^{k} F_i$.*
2. *For every point $p : X \to 1$ it holds that $|R^{\mathsf{T}}p| \leq k$.*

It states that every morphism can be represented by the union of pairwise disjoint univalent morphisms. As an immediate consequence we get that a morphism $R : X \to Y$ is univalent iff for every point $p : X \to 1$ it holds that $|R^{\mathsf{T}}p| \leq 1$. A dual characterisation can be obtained for total morphisms, i.e., that $R : X \to Y$ is total iff it holds $|R^{\mathsf{T}}p| \geq 1$

for every point $p : X \to 1$. Both statements are proved in [2]. Furthermore, in the context of graph theory, if $R : X \to X$ is an adjacency relation, the second property means that the maximum (in-)degree in the modelled graph is smaller than k. This theorem translates as follows in Coq:

```
Theorem decompose X Y (R: C X Y) k:
  (exists l: list (C X Y),
      length l = k ∧ disjoint l ∧
      (forall F, In F l → is_univalent l) ∧
      R == \sup_(F\in l) F)
  ↔ forall p: C X unit, is_point p → card (R^T*p) <== k.
```

As in the case of the point axiom, we use a list `l` to represent the collection $\{F_1, \ldots, F_k\}$ mentioned in Theorem 4.3. The predicate `disjoint l` states that any two distinct elements of the list `l` have an empty intersection.

The direct implication is easy; we obtain it in 5 lines. The reverse implication is proved by induction on k. The base case only requires 5 lines; the inductive case requires 23 lines, and an additional hypothesis on the considered relation algebra: a relational version of the *Axiom of Choice*. We use the following formulation, stemming from [23].

Definition 4.1 (Relational Axiom of Choice)**.** *A relation algebra satisfies the axiom of choice if for every morphism $R : X \to X$, there exists a univalent morphism F such that $F \subseteq R$ and $F\mathsf{L} = R\mathsf{L}$.*

In this example, although the resulting Coq proof closely follows the paper one, it is noticeably more compact. Our explanation is that once we move to relation algebraic proofs, convincing a computer becomes easier than convincing a human reader: we just need to name lemmas and axioms in the Coq proof, while we constantly need to recall the current algebraic expressions in a paper proof.

5 Conclusion

We presented an extension of the Coq RelationAlgebra library, that makes it possible to reason algebraically about cardinalities of binary relations. Using this library, we showed how to formalise results ranging from simple properties of linear orders or trees, to correctness of an approximation algorithm from graph theory, and a more abstract decomposition theorem.

A key feature of the Coq proof assistant for this work is *dependent types*: they allow us to define relations algebras as categories in a straightforward way, so that we can talk about vectors or units as one would do on paper. This sharply contrasts with the situation in other attempts with theorem provers, where the authors had to painfully restrict to homogeneous relations. Of course, such an restriction is not possible if one wants to deal with cardinals as in the present work.

The ability to mechanise basic results from graph theory in a streamlined way is something crucial from our point of view: proofs in graph theory are often quite informal and intuitive (for the sake of readability), so that their formalisation can require a lot of work. In the present work, we have shown that the relation algebraic approach could be an efficient way to attack such a difficult task.

References

[1] R. Berghammer, S. Bolus, A. Rusinowska, and H. C. M. de Swart. A relation-algebraic approach to simple games. *European Journal of Operational Research*, 210(1):68–80, 2011.

[2] R. Berghammer, N. Danilenko, P. Höfner, and I. Stucke. Cardinality of relations with applications. *Discrete Mathematics*, 339(12):3089–3115, 2016.

[3] R. Berghammer and S. Fischer. Combining relation algebra and data refinement to develop rectangle-based functional programs for reflexive-transitive closures. *Journal of Logical and Algebraic Methods in Programming*, 84(3):341–358, 2015.

[4] R. Berghammer, P. Höfner, and I. Stucke. Tool-Based Verification of a Relational Vertex Coloring Program. In Kahl et al. [18], pages 275–292.

[5] R. Berghammer, P. Höfner, and I. Stucke. Cardinality of Relations and Relational Approximation Algorithms. *Journal of Logical and Algebraic Methods in Programming*, 85(2):269–286, 2016.

[6] R. Berghammer and G. Struth. On Automated Program Construction and Verification. In C. Bolduc, J. Desharnais, and B. Ktari, editors, *Mathematics of Program Construction, MPC 2010*, volume 6120 of *LNCS*, pages 22–41. Springer, 2010.

[7] R. Berghammer, I. Stucke, and M. Winter. Investigating and computing bipartitions with algebraic means. In Kahl et al. [18], pages 257–274.

[8] Y. Bertot, G. Gonthier, S. O. Biha, and I. Pasca. Canonical big operators. In *TPHOLs*, volume 5170 of *LNCS*, pages 86–101. Springer, 2008.

[9] P. Brunet, D. Pous, and I. Stucke. Cardinalities of Finite Relations in Coq. In *Interactive Theorem Proving, ITP 2016, Proceedings*, volume 9807 of *LNCS*, pages 466–474. Springer, 2016.

[10] N. Danilenko. Using Relations to Develop a Haskell Program for Computing Maximum Bipartite Matchings. In W. Kahl and T. G. Griffin, editors, *Relational and Algebraic Methods in Computer Science, RAMiCS 2012*, volume 7560 of *LNCS*, pages 130–145. Springer, 2012.

[11] H. C. M. de Swart, R. Berghammer, and A. Rusinowska. Computational social choice using relation algebra and relview. In *Relations and Kleene Algebra in Computer Science, RelMiCS 2009*, pages 13–28, 2009.

[12] N. Francez. *Program Verification*. Addison-Wesley, 1992.

[13] H. Furusawa. *Algebraic Formalisations of Fuzzy Relations and Their Representation Theorems*. PhD thesis, Department of Informatics, Kyushu University, 1998.

[14] N. Galatos, P. Jipsen, T. Kowalski, and H. Ono. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. Elsevier, 2007.

[15] F. Garillot, G. Gonthier, A. Mahboubi, and L. Rideau. Packaging mathematical structures. In *TPHOL*, volume 5674 of *LNCS*, pages 327–342. Springer, 2009.

[16] D. Gries. *The Science of Programming*. Springer, 1987.

[17] P. Höfner and G. Struth. On Automating the Calculus of Relations. In A. Armando, P. Baumgartner, and G. Dowek, editors, *International Joint Conference Automated Reasoning, IJCAR 2008*, volume 5195 of *LNCS*, pages 50–66. Springer, 2008.

[18] W. Kahl, M. Winter, and J. N. Oliveira, editors. *Relational and Algebraic Methods in Computer Science - 15th International Conference, RAMiCS 2015, Braga, Portugal, September 28 - October 1, 2015, Proceedings*, volume 9348 of *LNCS*. Springer, 2015.

[19] Y. Kawahara. On the Cardinality of Relations. In R. A. Schmidt, editor, *Relations and Kleene Algebra in Computer Science, RelMiCS/AKA 2006*, volume 4136 of *LNCS*, pages 251–265. Springer, 2006.

[20] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.

[21] D. Pous. Relation Algebra and KAT in Coq. Website.

`http://perso.ens-lyon.fr/damien.pous/ra/`.

[22] D. Pous. Kleene Algebra with Tests and Coq tools for while programs. In *ITP*, volume 7998 of *LNCS*, pages 180–196. Springer, 2013.

[23] H. Rubin and J. E. Rubin. *Equivalents of the Axiom of Choice*. North-Holland, 1970.

[24] G. Schmidt and T. Ströhlein. Relation Algebras: Concept of Points and Representability. *Discrete Mathematics*, 54(1):83–92, 1985.

[25] G. Schmidt and T. Ströhlein. *Relations and Graphs - Discrete Mathematics for Computer Scientists*. EATCS Monographs on Theoretical Computer Science. Springer, 1993.

[26] A. Tarski. On the calculus of relations. *J. of Symbolic Logic*, 6(3):73–89, 1941.

[27] A. Tarski and S. Givant. *A Formalization of Set Theory without Variables*, volume 41 of *Colloquium Publications*. American Mathematical Society, Providence, Rhode Island, 1987.

[28] V. Wei. A Lower Bound for the Stability Number of a Simple Graph. *Bell Laboratories Technical Memorandum 81-11217-9*, 1981.