Verifying Minimum Spanning Tree Algorithms with Stone Relation Algebras

Walter Guttmann

Department of Computer Science and Software Engineering, University of Canterbury, New Zealand
walter.guttmann@canterbury.ac.nz

Abstract

We study a generalisation of relation algebras in which the underlying Boolean algebra structure is replaced with a Stone algebra. Many theorems of relation algebras generalise with no or small changes. Weighted graphs represented as matrices over extended real numbers form an instance. Relational concepts and methods can thus be applied to weighted graphs. We formally specify the problem of computing minimum spanning forests and express Kruskal's algorithm in relation-algebraic terms. We give a total-correctness proof of the algorithm. All results are formally verified in Isabelle/HOL. This paper is an extended version of [40].

Keywords: formal methods, Kleene algebras, relation algebras, Stone algebras, total correctness, weighted graphs

1. Introduction

Binary relations, which are the main instance of relation algebras, are essentially Boolean matrices. In graph theory they occur as the adjacency-matrix representation of unweighted graphs. It is therefore not surprising that relation-algebraic methods have been used to reason about graphs and to develop graph algorithms [63, 31, 12, 11, 10]. In this context weighted graphs are problematic simply because edge weights cannot be stored as entries of a Boolean matrix. Sometimes a workaround can be used, namely to represent weighted graphs by incidence matrices and weight functions [13]. However, keeping the direct representation of weighted graphs as matrices over numbers has benefits: it involves only one type of matrix, only a single matrix per graph, and only untyped (homogeneous) algebras which are better supported by theorem provers. Path problems and related algorithms have been treated successfully with this direct representation based on semirings with pre-orders and Kleene algebras [2, 7, 34, 44]. Other graph problems, in particular the minimum spanning tree problem, seem to require more structure. Relation algebras provide additional structure, but need to be generalised to capture weighted graphs.

In order to verify Prim's algorithm for minimum spanning trees, we have proposed such a generalisation, Stone relation algebras, in [37]. Edge weights are typically numbers and form lattice and semiring structures (such as max-min and min-plus algebras). However, they do not form a Boolean algebra because a complement operation cannot be defined on the underlying linear order of numbers. The idea is to generalise the Boolean algebra structure just so much that edge weights can be represented while most of the structure is preserved. In particular, edge weights support a pseudocomplement operation and even form a Stone algebra. In Stone algebras, the involution property $\overline{\overline{x}} = x$ and the law of excluded middle $x \sqcup \overline{x} = \top$ are missing, but the weaker $\overline{x} \sqcup \overline{\overline{x}} = \top$ still holds, as do De Morgan's laws and $x \sqcap \overline{x} = \bot$. By forming matrices over Stone algebras we can hope to preserve much of the structure of relations. These matrices represent weighted graphs and we capture their algebraic properties by Stone relation algebras. The axioms of Stone relation algebras are based on the axioms of Tarski's relation algebras, which are modified to account for the weakening of the underlying lattice structure from Boolean algebras to Stone algebras.

As a case study in this paper, we verify Kruskal's algorithm for computing minimum spanning trees [50]. It differs from Prim's algorithm by maintaining a spanning forest instead of a spanning tree during the computation. Since forests with more than one component do not have a unique root we can no longer use the specification given in [37], which involved the root of the constructed tree. Moreover, maintaining the forest property throughout Kruskal's algorithm is more involved than maintaining the tree property in Prim's algorithm. Nevertheless, most of the proof can be carried out in Stone-Kleene relation algebras, which combine Stone relation algebras with Kleene algebras in order to express reachability in graphs. As for Prim's algorithm, a small part of the correctness proof of Kruskal's algorithm relies on additional operations to obtain an edge with minimal weight and the sum of all edge weights in a graph. Axioms for these operations are based on [42]; we add axioms to solve two remaining issues: termination of the algorithm and existence of minimum spanning forests.

Our paper [37] gave the basic definitions and results with a focus on the verification of Prim's algorithm. In this paper, we study the properties of Stone relation algebras in more detail and verify the correctness of Kruskal's algorithm with respect to a modified specification. Related work is discussed throughout the present paper. Its structure and contributions are as follows:

- In Section 2 we study pseudocomplemented algebras in general, and Stone algebras in particular. We also discuss the extended-real and matrix models of Stone algebras. Many results in this section are known from the literature; we contribute formally verified proofs of the algebraic properties and of the instantiation to the models.
- In Section 3 we study Stone relation algebras. Our contribution is to show that many results of relation algebras generalise to Stone relation algebras directly or, in some cases, with small changes. This includes algebraic properties that hold for all elements and ones that hold for specific classes of elements. Again, we formally prove our results including the instantiation to models.
- In Section 4 we study the weighted-graph model of Stone relation algebras. Our contribution is to characterise in logical terms the meaning of relation-algebraic properties when applied to weighted graphs. Also here, our results are formally stated and proved.
- In Section 5 we study Stone-Kleene relation algebras, which extend Stone relation algebras with the Kleene star operation. We contribute a number of algebraic properties, again formally proved.
- In Section 6 we further study m-algebras, which support operations for finding minimal edges and for computing the total weight of a graph. The axioms are based on [37, 42]. Our contribution includes two new axioms which allow us to give a total-correctness proof of Kruskal's algorithm and dispose of an assumption about the existence of minimum spanning trees. We also prove that m-algebras satisfy the Tarski rule, which is required to deal with a conditional statement in Kruskal's algorithm.
- In Section 7 we formulate Kruskal's algorithm for computing minimum spanning forests in m-Kleene-algebras. We study the connected components of forests, based on which we specify the minimum spanning forest problem. We use an implementation of Kruskal's algorithm in Isabelle/HOL to show the existence of spanning forests in a constructive way. Based on this, we give a total-correctness proof of the algorithm using Hoare logic. We discuss key parts of the invariant that maintain the forest structure.

All of our results are verified in Isabelle/HOL [57] using its integrated automated theorem provers and SMT solvers [59, 17]. We use a library for Hoare logic with a tactic to generate verification conditions [54, 55], which we have extended from partial correctness to total correctness. We omit the proofs, which can be found in the theory files available in the Archive of Formal Proofs [38, 41, 39] and at http://www.csse.canterbury.ac.nz/walter.guttmann/algebra/. The archive currently stores the theories for Stone algebras, Stone relation algebras and Stone-Kleene relation algebras including the results presented in Sections 2–5. The theories for m-Kleene-algebras including the results of Sections 6 and 7 are being prepared for it.

The present paper is a version of [40] extended by Sections 6 and 7. Section 6 is based on previous work [37, 42]. We have modified Definition 17 by adding the new axioms (16) and (17), which we use to

overcome two limitations of the previous papers. Also new is a discussion of Tarski's rule for Stone relation algebras and Theorems 18 and 19. The latter generalises a known result of relation algebras, which is needed for Kruskal's algorithm but not for Prim's algorithm. Section 7 is entirely new. This includes a relation-algebraic formulation of Kruskal's algorithm and a modified specification to account for the construction of forests. Besides the technical challenge of reasoning about forests to prove that the various parts of the invariant are maintained, we use a new method of constructive reasoning about algorithms in Isabelle/HOL facilitated by total-correctness proofs. All new results, in particular Theorems 18–26 in Sections 6 and 7, have been formally proved in Isabelle/HOL requiring a substantial extension of the theories developed for [37, 40, 42].

2. Pseudocomplemented algebras

This section covers basic algebraic structures used in the present paper, including lattices, pseudocomplemented lattices and Stone algebras. These structures are further discussed in a number of textbooks [8, 15, 19, 24, 35]. Many results given in this section can be found in these textbooks. All results given in this section have been formally verified in Isabelle/HOL, mostly as part of a proof of Chen and Grätzer's construction theorem for Stone algebras [38].

Definition 1. *A* bounded semilattice *is an algebraic structure* $(S, \sqcup, \bot)$ *where* $\sqcup$ *is associative, commutative and idempotent and has unit* $\bot$*:*

$$x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z \qquad x \sqcup y = y \sqcup x \qquad x \sqcup x = x \qquad x \sqcup \bot = x$$

A bounded lattice *is an algebraic structure* $(S, \sqcup, \sqcap, \bot, \top)$ *where* $(S, \sqcup, \bot)$ *and* $(S, \sqcap, \top)$ *are bounded semilattices and the following absorption axioms hold:*

$$x \sqcup (x \sqcap y) = x \qquad x \sqcap (x \sqcup y) = x$$

A bounded distributive lattice *is a bounded lattice where the following distributivity axioms hold (it is enough to postulate one of the two to obtain the other):*

$$x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z) \qquad x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$$

The lattice order *is given by*

$$x \leq y \;\Leftrightarrow\; x \sqcup y = y$$

A (distributive) p-algebra *is an algebraic structure* $(S, \sqcup, \sqcap, \overline{\ }, \bot, \top)$ *such that* $(S, \sqcup, \sqcap, \bot, \top)$ *is a bounded (distributive) lattice and the pseudocomplement operation* $\overline{\ }$ *satisfies the equivalence*

$$x \sqcap y = \bot \;\Leftrightarrow\; x \leq \overline{y}$$

A Stone algebra *is a distributive p-algebra satisfying the equation*

$$\overline{x} \sqcup \overline{\overline{x}} = \top$$

An element $x \in S$ *is* regular *if* $\overline{\overline{x}} = x$ *and* dense *if* $\overline{x} = \bot$*. A* Boolean algebra *is a Stone algebra whose elements are all regular.*

Thus the pseudocomplement $\overline{y}$ of an element y is the $\leq$-greatest element whose meet with y is $\bot$. The following result gives basic properties of pseudocomplements.

Theorem 2. *Let* S *be a p-algebra and let* $x, y, z \in S$*. Then the operation* $\overline{\ }$ *is* $\leq$*-antitone,* $x \sqcup \overline{x}$ *is dense, and*

1. $\overline{\bot} = \top$
2. $\overline{\top} = \bot$
3. $x \leq \overline{\overline{x}}$
4. $\overline{\overline{\overline{x}}} = \overline{x}$
5. $x \leq \overline{y} \Leftrightarrow y \leq \overline{x}$
6. $x \sqcap y = \bot \Leftrightarrow \overline{\overline{x}} \sqcap y = \bot$
7. $x \sqcap y \leq \overline{z} \Leftrightarrow \overline{\overline{x}} \sqcap y \leq \overline{z}$
8. $x \sqcap y \leq \overline{z} \Leftrightarrow x \sqcap z \leq \overline{y}$
9. $x \sqcap \overline{x} = \bot$
10. $\overline{x \sqcup y} = \overline{x} \sqcap \overline{y}$
11. $\overline{\overline{\overline{x}} \sqcup \overline{\overline{y}}} = \overline{\overline{x \sqcup y}}$
12. $\overline{x \sqcap \overline{\overline{y}}} = \overline{x \sqcap y}$
13. $\overline{\overline{x \sqcap y}} = \overline{\overline{x}} \sqcap \overline{\overline{y}}$
14. $x \sqcap \overline{x \sqcap y} = x \sqcap \overline{y}$
15. $x \sqcap \overline{\overline{y}} \leq \overline{\overline{x \sqcap y}}$
16. $x \sqcup \overline{\overline{y}} \leq \overline{\overline{x \sqcup y}}$

In particular, the function $\lambda x.\overline{\overline{x}}$ is a closure operation, that is, idempotent, $\leq$-increasing and $\leq$-isotone. The image of the operation $\overline{\ }$ is precisely the set of regular elements. They are closed under the operations $\sqcap$, $\overline{\ }$, $\bot$ and $\top$. The dense elements of a p-algebra are precisely those mapped to $\top$ by the operation $\lambda x.\overline{\overline{x}}$. They are closed under the operations $\sqcup$, $\sqcap$, $\lambda x.\overline{\overline{x}}$ and $\top$. Equational axioms for p-algebras are obtained by adding Theorems 2.1, 2.2 and 2.14 to any set of equational axioms for bounded lattices.

In distributive p-algebras, we also obtain the following properties. By Theorem 3.1, every element x can be represented as the meet of a dense and a regular element.

Theorem 3. *Let S be a distributive p-algebra and let $x, y \in S$. Then*

1. $(x \sqcup \overline{x}) \sqcap \overline{\overline{x}} = x$
2. $x \sqcap y = \bot \wedge x \sqcup y = \top \Rightarrow \overline{x} = y$
3. $x \leq y \Leftrightarrow x \leq y \sqcup \overline{x}$
4. $x \leq y \Leftrightarrow x \sqcup \overline{x} \leq y \sqcup \overline{x}$

In a Stone algebra we obtain one of De Morgan's laws (the other is Theorem 2.10) and a number of weak shunting properties as the following result shows.

Theorem 4. *Let S be a Stone algebra and let $x, y, z \in S$. Then*

1. $\overline{x \sqcap y} = \overline{x} \sqcup \overline{y}$
2. $\overline{\overline{x \sqcup y}} = \overline{\overline{x}} \sqcup \overline{\overline{y}}$
3. $\overline{x \sqcup y} \sqcup \overline{x \sqcup \overline{y}} = \overline{x}$
4. $(x \sqcap \overline{y}) \sqcup (x \sqcap \overline{\overline{y}}) = x$
5. $\overline{\overline{x}} \sqcap y = \overline{\overline{x}} \sqcap z \wedge \overline{x} \sqcap y = \overline{x} \sqcap z \Rightarrow y = z$
6. $x \leq \overline{y} \Leftrightarrow \top = \overline{x} \sqcup \overline{y}$
7. $x \sqcap y \leq \overline{z} \Leftrightarrow x \leq \overline{z} \sqcup \overline{y}$
8. $x \sqcap \overline{y} \leq z \Leftrightarrow x \leq z \sqcup \overline{\overline{y}}$

The weak shunting property in Theorem 4.8 does not require the element z on the right-hand side to be regular. Another consequence is that the regular elements of a Stone algebra S are closed under the operation $\sqcup$, whence they form a Boolean subalgebra of S [35]. The dense elements of a Stone algebra form a distributive lattice with $\top$.

In the remainder of this section we look at instances of Stone algebras, notably extended real numbers and matrices over Stone algebras. Our considerations are motivated by weighted graphs. In this model we take edge weights from a Stone algebra and represent graphs by matrices containing edge weights.

For edge weights we use the extended real numbers $\mathbb{R}' = \mathbb{R} \cup \{\bot, \top\}$ with the operations max and min and the order $\leq$ extended so that $\bot$ is the $\leq$-least element and $\top$ is the $\leq$-greatest element. The resulting structure is a Stone algebra; the following result also shows the operation $\lambda x.\overline{\overline{x}}$ in this algebra.

Theorem 5. $(\mathbb{R}', \max, \min, \overline{\ }, \bot, \top)$ *is a Stone algebra with*

$$\overline{x} = \begin{cases} \top & \text{if } x = \bot \\ \bot & \text{if } x \neq \bot \end{cases} \qquad \overline{\overline{x}} = \begin{cases} \bot & \text{if } x = \bot \\ \top & \text{if } x \neq \bot \end{cases}$$

and the order $\leq$ on $\mathbb{R}'$ as the lattice order. The regular elements are $\bot$ and $\top$. All elements except $\bot$ are dense.

The operation $\lambda x.\overline{\overline{x}}$ checks whether its argument is different from $\bot$ and returns one of the Boolean elements $\bot$ or $\top$.

Weighted graphs are represented as matrices whose entries are edge weights. We therefore need to lift the Stone algebra structure to matrices according to the following result. Let $S^{A\times A}$ denote the set of square matrices with indices from a set A and entries from a set S. Such a matrix represents a directed graph with node set A and edge weights taken from S.

Theorem 6. *Let $(S,\sqcup,\sqcap,\overline{\ },\bot,\top)$ be a Stone algebra and let A be a set. Then $(S^{A\times A},\sqcup,\sqcap,\overline{\ },\bot,\top)$ is a Stone algebra, where the operations $\sqcup$, $\sqcap$, $\overline{\ }$, $\bot$, $\top$ and the lattice order $\leq$ are lifted componentwise.*

Using pointwise liftings, the result holds more generally for the set S^X of all functions from X to S, for any set X.

It follows that the regular elements among the matrices over extended reals are the matrices over $\{\bot,\top\}$. They represent unweighted graphs: an entry $M_{ij}=\bot$ means that there is no edge from node i to node j in graph M, while $M_{ij}=\top$ means that there is an edge but no information about its weight is provided. Hence, on the matrix level the operation $\lambda x.\overline{\overline{x}}$ takes a weighted graph and produces an unweighted graph. The result $\overline{\overline{M}}$ represents the structure of the weighted graph M after forgetting the weights. Under this interpretation, the dense elements among the matrices correspond to complete graphs.

There are several approaches related to obtaining the structure of a weighted graph. In [66] an operation that gives the least 'crisp' relation containing a fuzzy relation is discussed. In [25] the 'shape' is a relation that represents a superset of the non-zero entries of a matrix of complex numbers; an operation that gives the non-zero entries is not considered. In [51] the 'support' is an operation on matrices over natural numbers that maps 0 to 0 and each non-zero entry to 1. In [47] weighted graphs are represented by matrices over commutative semirings and their structure is obtained by a 'flattening' operation that maps each entry x to the smallest multiplicatively idempotent element whose product with x is x. The multiplicatively idempotent elements in $\mathbb{R}$ are 0 and 1.

We conclude this section with a brief comparison of Stone algebras and Heyting algebras, which are bounded lattices where all relative pseudocomplements exist. The pseudocomplement of an element y relative to an element z is the $\leq$-greatest element whose meet with y is below z. By specialising $z=\bot$ it follows that Heyting algebras form distributive p-algebras. A counterexample generated by Nitpick [18] witnesses that, in general, Heyting algebras do not form Stone algebras this way. The four-element Boolean algebra extended with a new greatest element is a Heyting algebra that is not a Stone algebra [19]. A counterexample given in [32, Example 4.6] shows that relative pseudocomplements need not exist in Stone algebras. Extended reals form a Heyting algebra where the pseudocomplement of y relative to z is $\top$ if $y\leq z$ and z otherwise; this extends to matrices using a pointwise lifting.

3. Stone relation algebras

In this section we further discuss the algebraic structure of matrices over Stone algebras. We have seen in Section 2 that such matrices form Stone algebras by lifting the operations componentwise. Moreover, they can be used to represent weighted graphs with edge weights taken from the extended reals. Finally, the subset of regular matrices obtained as the image of the closure operation $\lambda x.\overline{\overline{x}}$ represents unweighted graphs in this case. Because unweighted graphs correspond to relations, these observations suggest a generalisation of relation algebras to cover weighted graphs.

Definition 7. *A* Stone relation algebra *$(S,\sqcup,\sqcap,\cdot,\overline{\ },{}^{\mathsf{T}},\bot,\top,1)$ is a Stone algebra $(S,\sqcup,\sqcap,\overline{\ },\bot,\top)$ with a composition $\cdot$ and a converse ${}^{\mathsf{T}}$ and a constant 1 satisfying the following axioms (1)–(10). We abbreviate*

$x \cdot y$ as xy and let composition have higher precedence than the operators $\sqcup$ and $\sqcap$.

$$(xy)z = x(yz) \tag{1}$$
$$1x = x \tag{2}$$
$$(x \sqcup y)z = xz \sqcup yz \tag{3}$$
$$(xy)^{\mathsf{T}} = y^{\mathsf{T}}x^{\mathsf{T}} \tag{4}$$
$$(x \sqcup y)^{\mathsf{T}} = x^{\mathsf{T}} \sqcup y^{\mathsf{T}} \tag{5}$$
$$x^{\mathsf{T}^{\mathsf{T}}} = x \tag{6}$$
$$\bot x = \bot \tag{7}$$
$$xy \sqcap z \leq x(y \sqcap x^{\mathsf{T}}z) \tag{8}$$
$$\overline{\overline{x}y} = \overline{\overline{x}}\,\overline{\overline{y}} \tag{9}$$
$$\overline{\overline{1}} = 1 \tag{10}$$

A relation algebra $(S, \sqcup, \sqcap, \cdot, \overline{\ }, {}^{\mathsf{T}}, \bot, \top, 1)$ *is a Stone relation algebra whose reduct* $(S, \sqcup, \sqcap, \overline{\ }, \bot, \top)$ *is a Boolean algebra.*

An element $x \in S$ is a vector *if $x\top = x$, a* co-vector *if $\top x = x$,* reflexive *if $1 \leq x$,* co-reflexive *if $x \leq 1$,* irreflexive *if $x \leq \overline{1}$,* symmetric *if $x = x^{\mathsf{T}}$,* asymmetric *if $x \sqcap x^{\mathsf{T}} = \bot$,* antisymmetric *if $x \sqcap x^{\mathsf{T}} \leq 1$,* transitive *if $xx \leq x$,* univalent *if $x^{\mathsf{T}}x \leq 1$,* injective *if $xx^{\mathsf{T}} \leq 1$,* total *if $1 \leq xx^{\mathsf{T}}$,* surjective *if $1 \leq x^{\mathsf{T}}x$, a* mapping *if x is univalent and total,* bijective *if x is injective and surjective, a* point *if x is a bijective vector, and an* arc *if both $x\top$ and $x^{\mathsf{T}}\top$ are bijective.*

Tarski's relation algebras [65] require a Boolean algebra, axioms (1)–(6), and Theorem 8.20 below [52]. Axioms (7)–(10) follow from these properties. Another way to obtain relation algebras is by requiring a Boolean algebra and axioms (1)–(8) since axioms (9) and (10) immediately follow in Boolean algebras. There is a large body of research about Tarski's relation algebras; recent monographs are [43, 53]. See [4] for an implementation of Tarski's relation algebras in Isabelle/HOL.

The Dedekind formula (8) or variants of it are known from [63, 30, 14, 46]. In particular, Dedekind categories algebraically capture fuzzy relations, which are matrices over the real unit interval or complete distributive lattices used for modelling fuzzy systems [33, 66]. In Dedekind categories composition is required to have a left residual and each Hom-set must be a complete distributive lattice and therefore a Heyting algebra [45]. Stone relation algebras maintain the signature of relation algebras. Algebras of relations with a smaller signature have been studied, for example, in [20, 3]. In rough relation algebras [21, 28] the lattice structure is required to be a double Stone algebra, which involves two dual pseudocomplement operations. A rough relation is a pair of upper and lower approximations of a relation with respect to a fixed indiscernibility relation [60]. Double residuated lattices for a fuzzy generalisation of information systems have been considered in [58, 27].

Regular elements are closed under composition and its unit by axioms (9) and (10).

The following properties hold in Stone relation algebras.

Theorem 8. *Let S be a Stone relation algebra and let $w, x, y, z \in S$. Then*

1. ${}^{\mathsf{T}}$ *and* $\cdot$ *are* $\leq$*-isotone*
2. $\bot^{\mathsf{T}} = \bot$
3. $\top^{\mathsf{T}} = \top$
4. $1^{\mathsf{T}} = 1$
5. $(x \sqcap y)^{\mathsf{T}} = x^{\mathsf{T}} \sqcap y^{\mathsf{T}}$
6. $\overline{x}^{\mathsf{T}} = \overline{x^{\mathsf{T}}}$
7. $x\bot = \bot$
8. $x1 = 1$
9. $x \leq x\top$
10. $x \leq \top x$
11. $\top\top = \top$
12. $x \leq xx^{\mathsf{T}}x$
13. $x\top x\top = x\top$
14. $x(y \sqcup z) = xy \sqcup xz$
15. $x(y \sqcap z) \leq xy \sqcap xz$
16. $(x \sqcap y)z \leq xz \sqcap yz$

17. $x = (1 \sqcap xx^\mathsf{T})x = x(1 \sqcap x^\mathsf{T}x)$
18. $yx \sqcap z \leq (y \sqcap zx^\mathsf{T})x$
19. $xy \sqcap z = (x \sqcap zy^\mathsf{T})(y \sqcap x^\mathsf{T}z) \sqcap z$
20. $x^\mathsf{T}\overline{xy} \leq \overline{y}$
21. $xy \leq \overline{z} \Leftrightarrow x^\mathsf{T}z \leq \overline{y}$
22. $xy \leq \overline{z} \Leftrightarrow zy^\mathsf{T} \leq \overline{x}$
23. $xy \leq \overline{z} \Leftrightarrow yz^\mathsf{T} \leq \overline{x}^\mathsf{T}$
24. $xy \leq \overline{z} \Leftrightarrow z^\mathsf{T}x \leq \overline{y}^\mathsf{T}$
25. $xyz \leq \overline{w} \Leftrightarrow x^\mathsf{T}wz^\mathsf{T} \leq \overline{y}$
26. $xy \leq \overline{1} \Leftrightarrow yx \leq \overline{1}$
27. $x\overline{\overline{y}} \leq \overline{\overline{xy}}$
28. $\overline{x\overline{\overline{y}}} = \overline{xy}$
29. $\overline{\overline{x}y} \leq \overline{\overline{xy}}$
30. $\overline{\overline{\overline{x}y}} = \overline{xy}$
31. $xy \sqcap \overline{xz} = x(y \sqcap \overline{z}) \sqcap \overline{xz}$
32. $\overline{xy} \sqcup \overline{\overline{xz}} = \overline{x(y \sqcap \overline{z})} \sqcup \overline{\overline{xz}}$

Theorems 8.21–8.24 are weak versions of the Schröder equivalences of relation algebras: the right-hand sides of both inequalities must be regular, which is achieved by applying the pseudocomplement. On the other hand, the conjugation property

$$xy \sqcap z = \bot \;\Leftrightarrow\; y \sqcap x^\mathsf{T}z = \bot \;\Leftrightarrow\; x \sqcap zy^\mathsf{T} = \bot$$

also holds in Stone relation algebras. Theorem 8.32 is another example how a property of relation algebras has been weakened, in this case by introducing double pseudocomplements. The original version is $\overline{xy} \sqcup xz = \overline{x(y \sqcap \overline{z})} \sqcup xz$ and appears as [52, Theorem 24(xxiv)].

Counterexamples generated by Nitpick witness that neither the Schröder equivalences of relation algebras nor [52, Theorem 24(xxiv)] hold in Stone relation algebras. Nevertheless, Theorem 8 shows that many properties of relation algebras already hold in Stone relation algebras.

We reuse the characterisations of vectors, co-reflexivity, injectivity and other properties known from relation algebras [63]. In relation algebras, arcs are also known as atoms since they are atoms in the sense of lattice theory, which is not the case in Stone relation algebras. Consequences of these definitions are given by the following result. Once again, it shows that many properties generalise from relation algebras without changes.

Theorem 9. *Let S be a Stone relation algebra and let $w, x, y, z \in S$.*

1. *The regular elements of S are closed under the operation $^\mathsf{T}$.*
2. *The set of vectors of S is closed under the operations $\sqcup$, $\sqcap$, $\cdot$, $^{\overline{\ }}$, $\bot$ and $\top$.*
3. *Every mapping, every bijective element and every arc is regular.*

If w and x are vectors, then

4. $(x \sqcap y)z = x \sqcap yz$
5. $y(z \sqcap x^\mathsf{T}) = yz \sqcap x^\mathsf{T}$
6. $(y \sqcap x^\mathsf{T})z = (y \sqcap x^\mathsf{T})(x \sqcap z)$
7. $(y \sqcap x^\mathsf{T})z = y(x \sqcap z)$
8. *yx is a vector*
9. $\overline{x}^\mathsf{T}x = \bot$
10. $xx^\mathsf{T}xx^\mathsf{T} \leq xx^\mathsf{T}$
11. $wx^\mathsf{T} = w \sqcap x^\mathsf{T}$

If w and x are co-reflexive, then

12. $x^\mathsf{T} = x$
13. $x\top \sqcap y = xy$
14. $x\top \sqcap 1 = x$
15. $\overline{x\top} \sqcap 1 = \overline{x} \sqcap 1$
16. $xx = x$
17. $xy \sqcap \overline{z} = xy \sqcap \overline{xz}$
18. $w \sqcap x = wx$
19. $wy \sqcap xy = (w \sqcap x)y$

If w is univalent and x is injective, then

20. $w(y \sqcap z) = wy \sqcap wz$
21. $w\overline{y} \leq \overline{wy}$
22. $(y \sqcap z)x = yx \sqcap zx$
23. $\overline{y}x \leq \overline{yx}$

If w is a mapping and x is bijective, then

24. $y \leq wz \Leftrightarrow w^\mathsf{T} y \leq z$
25. $w\overline{y} = \overline{wy}$
26. $y \leq zx \Leftrightarrow yx^\mathsf{T} \leq z$
27. $\overline{y}x = \overline{yx}$

Finally,

28. *x is a vector/univalent/total $\Leftrightarrow$ x^T is a co-vector/injective/surjective*
29. *x is total $\Leftrightarrow$ $x\top = \top$*
30. *x is surjective $\Leftrightarrow$ $\top x = \top$*

Many properties already hold in more general structures than Stone relation algebras. For example, consider single-object bounded distributive allegories [30]. They are obtained by expanding bounded distributive lattices with a composition, a converse and a unit satisfying axioms (1)–(8). All statements of Theorems 8 and 9 that do not involve the pseudocomplement or properties defined using the pseudocomplement already follow in this setting. Going further, expand single-object bounded distributive allegories by a pseudocomplement operation such that they form (distributive) p-algebras. All statements of Theorems 8 and 9 except Theorems 8.32 and 9.3 follow in this setting.

While vectors are closed under pseudocomplements in Stone relation algebras, a counterexample generated by Nitpick witnesses that x need not be a vector if $\overline{x}$ is a vector. In fact there are counterexamples in the weighted-graph model as shown below.

In order to instantiate Stone relation algebras by weighted graphs we proceed in two steps. First, we show how every Stone algebra gives rise to a Stone relation algebra by reusing some of the operations. Second, we lift the Stone relation algebra structure to matrices; this is similar to the lifting for Dedekind categories [66]. The following result also shows that every Stone relation algebra has a subalgebra that is a relation algebra. As a consequence we can work with weighted graphs in Stone relation algebras and use the full power of relation algebras for reasoning about their structure.

Theorem 10.

1. *The regular elements of a Stone relation algebra S form a relation algebra that is a subalgebra of S.*
2. *Let $(S, \sqcup, \sqcap, \overline{\ }, \bot, \top)$ be a Stone algebra. Then $(S, \sqcup, \sqcap, \sqcap, \overline{\ }, \lambda x.x, \bot, \top, \top)$ is a Stone relation algebra with meet as composition, $\top$ as its unit, and the identity function as converse.*
3. *Let $(S, \sqcup, \sqcap, \cdot, \overline{\ }, ^\mathsf{T}, \bot, \top, 1)$ be a Stone relation algebra and let A be a finite set. Then the structure $(S^{A\times A}, \sqcup, \sqcap, \cdot, \overline{\ }, ^\mathsf{T}, \bot, \top, 1)$ is a Stone relation algebra, where the operations $\cdot$, $^\mathsf{T}$ and 1 are defined by*

$$(M \cdot N)_{ij} = \bigsqcup_{k\in A} M_{ik} \cdot N_{kj}$$
$$(M^\mathsf{T})_{ij} = (M_{ji})^\mathsf{T}$$
$$1_{ij} = \begin{cases} 1 & \text{if } i = j \\ \bot & \text{if } i \neq j \end{cases}$$

The weighted-graph model is an instance of this construction because edge weights are taken from the Stone algebra of extended reals. Thus for a finite set A, the set of matrices $\mathbb{R}'^{A\times A}$ is a Stone relation algebra with the following operations:

$$(M \cdot N)_{ij} = \max_{k\in A} \min\{M_{ik}, N_{kj}\}$$
$$(M^\mathsf{T})_{ij} = M_{ji}$$
$$1_{ij} = \begin{cases} \top & \text{if } i = j \\ \bot & \text{if } i \neq j \end{cases}$$

The remaining operations are lifted componentwise from the underlying Stone algebra.

Recall that the regular elements are the matrices over $\{\bot, \top\}$ in this case; they represent unweighted graphs. In particular, the graphs $\bot$, 1 and $\top$ are regular. Theorem 10.1 confirms that these matrices form a relation algebra.

We furthermore note that the way to obtain regular matrices (essentially relations) from weighted matrices by taking the image of $\overline{\ }$ is similar to the way co-reflexive relations are obtained from relations by taking the image of the antidomain operation [26]. The operation $\lambda x.\overline{\overline{x}}$ corresponds to the domain operation and, if vectors are used instead of co-reflexives, to the operation $\lambda x.x\top$, which is a closure operation.

Next, we further discuss the difference between relation algebras and Stone relation algebras. The following list shows a number of properties of relation algebras that do not generally hold in Stone relation algebras. We give counterexamples found by Nitpick in the weighted-graph model of matrices over extended reals $\mathbb{R}'^{A\times A}$. Nitpick allows the user to set independent bounds for the size of matrices and the size of the set that approximates matrix entries in the search.

1. $xy \leq z \Leftrightarrow x^\mathsf{T}\overline{z} \leq \overline{y}$:
   This Schröder equivalence fails for $A = \{a\}$ and $x_{aa} = y_{aa} = 0$ and $z_{aa} = -1$.
2. $\overline{xy} \sqcup xz = \overline{x(y \sqcap \overline{z})} \sqcup xz$:
   This equation is [52, Theorem 24(xxiv)] and fails for $A = \{a\}$ and $x_{aa} = z_{aa} = 0$ and $y_{aa} = \top$.
3. $\overline{\overline{x}\top} \sqcap 1 \leq x$ for each vector x:
   This fails for $A = \{a\}$ and $x_{aa} = 0$.
4. $(\overline{\overline{x}\top} \sqcap 1)\top = x$ for each vector x:
   This fails for $A = \{a\}$ and $x_{aa} = 0$.
5. x is a vector if $\overline{x}$ is a vector:
   This holds for graphs with a single node but fails for $A = \{a, b\}$ and $x_{aa} = 0$ and $x_{ab} = x_{bb} = 1$ and $x_{ba} = \top$.
6. $\overline{x} \sqcap y$ is regular for each $x \neq \bot$:
   This holds for graphs with a single node but fails for $A = \{a, b\}$ and $x_{ba} = \bot$ and $y_{ba} = 1$ and all other entries of x and y set to $\top$.

Except in the last case, Nitpick indicated that the examples it found are 'potentially spurious', which might be due to the involved matrix products. All counterexamples have been verified manually.

Finally, we discuss two examples for proving properties of weighted graphs in Stone relation algebras. First, consider a graph G on a set of nodes A and a subset $B \subseteq A$ of the nodes. We work in the Stone relation algebra $S = \mathbb{R}'^{A\times A}$. The graph G is represented by an element $x \in S$ and the subset B of nodes is represented by a regular vector $v \in S$. The element vv^T describes the complete unweighted graph formed by the nodes in B. The meet $vv^\mathsf{T} \sqcap x$ restricts the edges of G to those that start and end in B; this is a weighted subgraph. By Theorem 9.10 we obtain

$$(vv^\mathsf{T} \sqcap x)(vv^\mathsf{T} \sqcap x) \leq vv^\mathsf{T}vv^\mathsf{T} \leq vv^\mathsf{T}$$

which shows that by following a sequence of two edges in the weighted subgraph we cannot leave the set of nodes in B. The claim extends to longer sequences of edges by using the Kleene star as in Section 5. Results like this are used for reasoning about Prim's minimum spanning tree algorithm, where in each step the constructed tree is a subgraph of the input and a spanning tree of the nodes that have already been visited.

Second, in the same setting let $e \in S$ such that $e \leq v\overline{v}^\mathsf{T}$. Such an e can represent a set of edges each of which goes from a node in B to a node outside of B. By Theorem 9.9 we obtain

$$ee \leq v\overline{v}^\mathsf{T}v\overline{v}^\mathsf{T} = \bot$$

which shows that it is not possible to follow two such edges in sequence. In Prim's algorithm, the edges considered for extending the spanning tree in each step satisfy this property. The obtained result is used for showing that the extended tree is acyclic.

4. Relational properties of weighted graphs

In this section we study the weighted-graph model of Stone relation algebras. In particular, we discuss how relation-algebraic properties are interpreted in this instance. Throughout this section, a graph is an element of the Stone relation algebra $\mathbb{R}'^{A\times A}$ introduced in Section 3.

4.1. Mappings and related properties

We first look at univalent, injective, total, surjective and bijective matrices and at mappings.

Theorem 11. *Let $M \in \mathbb{R}'^{A\times A}$. Then M is*

1. *univalent $\Leftrightarrow$ in every row at most one entry is not $\bot$*
2. *injective $\Leftrightarrow$ in every column at most one entry is not $\bot$*
3. *total $\Leftrightarrow$ in every row at least one entry is $\top$*
4. *surjective $\Leftrightarrow$ in every column at least one entry is $\top$*
5. *a mapping $\Leftrightarrow$ in every row exactly one entry is $\top$ and the others are $\bot$*
6. *bijective $\Leftrightarrow$ in every column exactly one entry is $\top$ and the others are $\bot$*

Moreover,

7. $\overline{M\top} = \bot \Leftrightarrow$ *in every row at least one entry is not $\bot$*
8. $\top\overline{M} = \bot \Leftrightarrow$ *in every column at least one entry is not $\bot$*

Note that univalent, injective, total and surjective matrices may have entries which are neither $\bot$ nor $\top$. In the graph interpretation, univalent means that every node has at most one outgoing edge. To specify at least one outgoing edge, we can use the property in Theorem 11.7, which is equivalent to $\overline{\overline{M}}$ being total. Requiring M to be total is stronger: it means that at least one edge is labelled with $\top$. Therefore, to specify exactly one outgoing edge per node, the conjunction of univalent with the property in Theorem 11.7 has to be used. Requiring a mapping is more restrictive; in fact mappings are regular by Theorem 9.3. Similar remarks apply for injective, surjective and bijective matrices and the property in Theorem 11.8 with respect to the incoming edges of each node.

4.2. Vectors and related properties

We next look at vectors, co-vectors, points and arcs in the weighted-graph model of matrices over $\mathbb{R}'$.

Theorem 12. *Let $M \in \mathbb{R}'^{A\times A}$. Then M is*

1. *a vector $\Leftrightarrow$ in every row all entries are the same*
2. *a co-vector $\Leftrightarrow$ in every column all entries are the same*
3. *a point $\Leftrightarrow$ exactly one row is constant $\top$ and the others are constant $\bot$*
4. *an arc $\Leftrightarrow$ exactly one entry is $\top$ and the others are $\bot$*

Also vectors and co-vectors may have entries which are neither $\bot$ nor $\top$. As in the relational case, a matrix with just one column/row is sufficient to store the information contained in a vector/co-vector. Points and arcs are regular by Theorem 9.3. Their interpretation for graphs is the same as in the relational model: a point represents a node of the graph and an arc represents an edge. Weaker properties can again be obtained by replacing surjective with the property in Theorem 11.8 in the definitions of point and arc. In this case, all rows in a point would be $\bot$ except for one row, in which all entries would be the same, arbitrary non-$\bot$ value. Similarly, an arc would have exactly one non-$\bot$ value.

4.3. Orders and related properties

Finally, we look at reflexive, co-reflexive, irreflexive, symmetric, antisymmetric, asymmetric and transitive matrices over $\mathbb{R}'$.

Theorem 13. *Let $M \in \mathbb{R}'^{A\times A}$. Then M is*

1. *reflexive $\Leftrightarrow$ the diagonal is constant $\top$*
2. *co-reflexive $\Leftrightarrow$ all entries not on the diagonal are $\bot$*
3. *irreflexive $\Leftrightarrow$ the diagonal is constant $\bot$*
4. *symmetric $\Leftrightarrow$ $M_{ij} = M_{ji}$ for each $i, j \in A$*
5. *antisymmetric $\Leftrightarrow$ $M_{ij} = \bot$ or $M_{ji} = \bot$ for each $i \neq j \in A$*
6. *asymmetric $\Leftrightarrow$ $M_{ij} = \bot$ or $M_{ji} = \bot$ for each $i, j \in A$*
7. *transitive $\Leftrightarrow$ $M_{ik} \leq M_{ij}$ or $M_{kj} \leq M_{ij}$ for each $i, j, k \in A$*

Co-reflexive matrices share most properties of tests [49, 36] except they do not form a Boolean algebra. Nevertheless, they form a Stone relation subalgebra in which composition and meet coincide and composition is idempotent. For example, the composition MN of a co-reflexive matrix M and an arbitrary matrix N restricts the elements of N in row i to at most M_{ii}. In particular, rows are filtered out if $M_{ii} = \bot$ and left unchanged if $M_{ii} = \top$. The composition NM has a similar effect on the columns of N.

Matrices that are reflexive, transitive and symmetric have a block-diagonal structure (that is, the base set A can be suitably partitioned by an equivalence relation). The entries in each block are different from $\bot$ but not necessarily $\top$.

Similarly, matrices that are reflexive, transitive and antisymmetric have the structure of a partial order. Again the non-$\bot$ entries may differ from $\top$. In the graph interpretation, antisymmetric means that there is at most one edge between any two different nodes. Asymmetric additionally requires that there are no loops; the latter property amounts to being irreflexive. Symmetric matrices can be used to represent undirected weighted graphs.

5. Stone-Kleene relation algebras

In this section we discuss iterated composition in Stone relation algebras. This works analogously to adding the Kleene star operation to relation algebras. In the graph model, this allows us to talk about reachability. We use the axioms of the Kleene star given in [48].

Definition 14. *A* Stone-Kleene relation algebra $(S, \sqcup, \sqcap, \cdot, ^-, ^\mathsf{T}, ^*, \bot, \top, 1)$ *expands a Stone relation algebra* $(S, \sqcup, \sqcap, \cdot, ^-, ^\mathsf{T}, \bot, \top, 1)$ *with an operation * satisfying the unfold and induction axioms*

$$\begin{array}{ll} 1 \sqcup yy^* \leq y^* & z \sqcup yx \leq x \Rightarrow y^*z \leq x \\ 1 \sqcup y^*y \leq y^* & z \sqcup xy \leq x \Rightarrow zy^* \leq x \end{array}$$

and the axiom

$$\overline{\overline{x^*}} = \overline{\overline{x}}^* \tag{11}$$

An element $x \in S$ is acyclic *if xx^* is irreflexive, and x is a* forest *if x is injective and acyclic.*

Kleene algebras are based on idempotent semirings in [48], but we do not require more axioms than the above since all Stone relation algebras are idempotent semirings. Regular elements are closed under the Kleene star by axiom (11). The following properties hold in Stone-Kleene relation algebras.

Theorem 15. *Let S be a Stone-Kleene relation algebra and let $x, y \in S$.*

1. *The regular elements of S are closed under the operation *.*

Moreover

2. $x^{*\mathsf{T}} = x^{\mathsf{T}*}$
3. $x^{\mathsf{T}}(xx^{\mathsf{T}})^* \leq x^{\mathsf{T}}$ *if* x *is a vector*
4. $(xx^{\mathsf{T}})^* = 1 \sqcup xx^{\mathsf{T}}$ *if* x *is a vector*
5. $x^{\mathsf{T}}y^* = x^{\mathsf{T}}((x^{\mathsf{T}}y^*)^{\mathsf{T}}(x^{\mathsf{T}}y^*) \sqcap y)^*$ *if* x *is a vector*
6. $x^{\mathsf{T}*} \leq \overline{x}$ *if and only if* x *is acyclic*
7. x *is asymmetric if* x *is acyclic*
8. $x^*x^{\mathsf{T}*} \sqcap x^{\mathsf{T}}x \leq 1$ *if* x *is a forest*

As an example we discuss Theorem 15.5, which considers a graph y and a set of nodes x represented as a vector. Then $y^{\mathsf{T}*}x$ is a vector representing the set of nodes reachable from any node in x. The same set is represented by the left-hand side $x^{\mathsf{T}}y^*$ as a co-vector. The right-hand side uses the same construction except the graph y is restricted to those edges that start and end in this set of reachable nodes. Thus Theorem 15.5 states that to reach any of these nodes from x it suffices to take edges between these nodes. This property is used several times for proving the correctness of Prim's minimum spanning tree algorithm.

As another example, Theorem 15.6 has the following interpretation for an acyclic graph x. The left-hand side describes backward reachability in x. The inequality states that if a node q is reachable from a node p by going backward any number of steps in x, then there must not be an edge from p to q; otherwise we could combine it with the path from q to p to obtain a cycle in x. Moreover, this condition is equivalent to being acyclic.

We finally note that a number of facts such as Theorem 15.8 can also be interpreted in rewrite systems similarly to Church-Rosser properties [64].

Again, structures more general than Stone-Kleene relation algebras can be studied. For example, consider single-object bounded distributive allegories expanded by an operation * satisfying the Kleene algebra axioms. All statements of Theorem 15 that do not involve the pseudocomplement or properties defined using the pseudocomplement already follow in this setting. Going further, expand these structures by a pseudocomplement operation such that they form (distributive) p-algebras. All statements of Theorem 15 except Theorem 15.1 follow in this setting.

In order to instantiate Stone-Kleene relation algebras by weighted graphs we extend the two-step process we used for Stone relation algebras in Section 3 by the Kleene star operation. First, every Stone algebra gives rise to a Stone-Kleene relation algebra by setting $x^* = \top$. This is because the underlying bounded lattice forms a semiring where $1 = \top$. Second, we lift the Stone-Kleene relation algebra structure to matrices. Note that $x^* = \top$ does not generally hold in the matrix algebra; only the entries on the diagonal of x^* will be $\top$.

Theorem 16.

1. *Let* $(S, \sqcup, \sqcap, ^{-}, \bot, \top)$ *be a Stone algebra. Then, using the constant* $\top$ *function as the Kleene star,* $(S, \sqcup, \sqcap, \sqcap, ^{-}, \lambda x.x, \lambda x.\top, \bot, \top, \top)$ *is a Stone-Kleene relation algebra.*
2. *Let* $(S, \sqcup, \sqcap, \cdot, ^{-}, ^{\mathsf{T}}, ^*, \bot, \top, 1)$ *be a Stone-Kleene relation algebra and let* A *be a finite set. Then* $(S^{A\times A}, \sqcup, \sqcap, \cdot, ^{-}, ^{\mathsf{T}}, ^*, \bot, \top, 1)$ *is a Stone-Kleene relation algebra, where the operation* * *on matrices is defined using Conway's automata-based construction described in [22, 48].*

The subalgebra of regular elements of a Stone-Kleene relation algebra is both a relation algebra and a Kleene algebra.

The proof of Theorem 16 formally verifies the correctness of Conway's construction for Kleene algebras. An implementation of the construction in Isabelle/HOL that extends [5] was given in [6] without a correctness proof.

6. Algebras for summing and minimising weights

In this section we add operations for summing and minimising edge weights to Stone-Kleene relation algebras. They are used in the correctness proof of minimum spanning tree algorithms.

- The operation m finds an edge that has minimal weight among all edges in a graph.

- The operation s gives the sum of the edge weights in a graph.
- The operation $+$ adds the weights of edges of two graphs on the same node set.

These operations have been axiomatised in [37] for verifying the correctness of Prim's minimum spanning tree algorithm. The axioms have been simplified in [42] using Theorem 9.3 of the present paper. The same operations and axioms can be used for a partial-correctness proof of Kruskal's algorithm. Axiom (12) has been slightly specialised in the present paper. We add the two new axioms (16) and (17) to obtain a total-correctness proof.

Definition 17. *An* s-algebra $(S, \sqcup, \sqcap, \cdot, +, \overline{\ }, {}^{\mathsf{T}}, s, \bot, \top, 1)$ *is a Stone relation algebra* $(S, \sqcup, \sqcap, \cdot, \overline{\ }, {}^{\mathsf{T}}, \bot, \top, 1)$ *with an addition* $+$ *and a summation* s *satisfying the following properties:*

$$x \neq \bot \wedge s(x) \leq s(y) \Rightarrow s(z) + s(x) \leq s(z) + s(y) \tag{12}$$
$$s(x) + s(\bot) = s(x) \tag{13}$$
$$s(x) + s(y) = s(x \sqcup y) + s(x \sqcap y) \tag{14}$$
$$s(x^{\mathsf{T}}) = s(x) \tag{15}$$

A linear s-algebra *is an s-algebra* S *satisfying the following properties:*

$$s(x) \leq s(y) \vee s(y) \leq s(x) \tag{16}$$
$$\{\overline{x} \mid x \in S\} \text{ is finite} \tag{17}$$

An m-algebra $(S, \sqcup, \sqcap, \cdot, +, \overline{\ }, {}^{\mathsf{T}}, s, m, \bot, \top, 1)$ *is a linear s-algebra* $(S, \sqcup, \sqcap, \cdot, +, \overline{\ }, {}^{\mathsf{T}}, s, \bot, \top, 1)$ *with a minimum selection* m *satisfying the following properties:*

$$m(x) \leq \overline{\overline{x}} \tag{18}$$
$$x \neq \bot \Rightarrow m(x) \text{ is an arc} \tag{19}$$
$$y \text{ is an arc} \wedge y \sqcap x \neq \bot \Rightarrow s(m(x) \sqcap x) \leq s(y \sqcap x) \tag{20}$$

An m-Kleene-algebra *is a structure* $(S, \sqcup, \sqcap, \cdot, +, \overline{\ }, {}^{\mathsf{T}}, {}^{*}, s, m, \bot, \top, 1)$ *such that* $(S, \sqcup, \sqcap, \cdot, \overline{\ }, {}^{\mathsf{T}}, {}^{*}, \bot, \top, 1)$ *is a Stone-Kleene relation algebra and* $(S, \sqcup, \sqcap, \cdot, +, \overline{\ }, {}^{\mathsf{T}}, s, m, \bot, \top, 1)$ *is an m-algebra.*

Axiom (16) requires that sums of edge weights are linearly ordered. This does not follow in s-algebras, which support also partial orders. However, it is a natural property when unique minimal edge weights are required, such as for both Prim's and Kruskal's algorithms. See [16] for extensions of minimum spanning tree algorithms to partially ordered edge weights. A counterexample generated by Nitpick witnesses that axiom (16) is independent of the other axioms of m-algebras.

Axiom (17) states that there are only finitely many regular elements. Pseudocomplemented algebras in which the set of regular elements (also called 'center') is finite have been studied, for example, in [9, 62]. In the weighted graph model, regular elements correspond to unweighted graphs, so this axiom states that there are finitely many unweighted graphs (for any given set of nodes), which ensures that the number of edges and the number of nodes is finite. Again, this is a natural property in the context of m-algebras to obtain unique minimal edge weights. Note that the model still admits infinitely many different edge weights and therefore infinitely many weighted graphs. An independence check for axiom (17) lies outside the scope of Nitpick as it would require an infinite counterexample.

Prim's and Kruskal's algorithms use the operation m to select an edge with minimal weight. The correctness specifications of these algorithms involve the operation s to describe the sum of edge weights, which needs to be minimised. The operation $+$ is only used in the axioms of s, neither in the algorithm nor in its specification.

The remaining axioms have the following meaning [42]:

(12) The operation $+$ is $\leq$-isotone in its second argument on the image of the operation s for non-empty graphs (this is required because edges may have negative weights).

(13) The empty graph adds no weight; the given axiom is weaker than the conjunction of $s(\bot) = \bot$ and $x + \bot = x$.

(14) This generalises the inclusion-exclusion principle to sets of weights.

(15) Reversing edges does not change the sum of weights.

(18) Ignoring weights, the minimal edge is contained in the graph.

(19) The result of m is just one unweighted edge, if the graph is not empty.

(20) Any edge y in the graph x weighs at least as much as $m(x)$; the operation s is used to compare the weights of edges between different nodes.

See [42] for consequences of these s-algebra and m-algebra axioms. We add a new result, which shows that a version of the Tarski rule of relation algebras holds in m-algebras. A Stone relation algebra satisfies the Tarski rule if

$$x = \overline{\overline{x}} \wedge x \neq \bot \Rightarrow \top x \top = \top \tag{21}$$

The additional condition $x = \overline{\overline{x}}$ is not needed in relation algebras since all elements are regular there. The following result follows from axioms (18) and (19) without using the other axioms.

Theorem 18. *Every m-algebra satisfies the Tarski rule (21).*

This holds since $m(x)$ is an arc by (19) and an assumption of (21), whence $\top m(x) \top = \top$ by Theorem 9.30, which implies the claim since $m(x) \leq x$ by (18) and the other assumption of (21).

In relation algebras, the Tarski rule implies that an arc is contained in an arbitrary element or its complement. This generalises to Stone relation algebras as follows.

Theorem 19. *Let S be a Stone relation algebra satisfying the Tarski rule and let $x, y \in S$ such that x is an arc. Then $x \leq \overline{y}$ or $x \leq \overline{\overline{y}}$.*

The main consequence of this result in the present work is that it allows us to distinguish whether the edge selected by Kruskal's algorithm connects two nodes in the same component of the constructed forest or two different components. This choice needs to be made in a conditional statement of Kruskal's algorithm, since it adds edges only between different components. Specifically, the algorithm will use the condition $e \leq \overline{c(f)}$ that holds if the selected edge e is outside of the components $c(f)$ of the forest f. If the condition does not hold, Theorem 19 and properties of e and f let us conclude $e \leq c(f)$, which is used in the Hoare-logic proof to show that the invariant is preserved also in this case. The while-loop in Prim's algorithm does not contain a conditional statement and therefore the Tarski rule was not required to prove its correctness.

Section 7 shows that the axioms of m-Kleene-algebras are sufficient to prove the correctness of Kruskal's minimum spanning tree algorithm. They are also sufficient to prove the correctness of Prim's minimum spanning tree algorithm [37]. In particular, the minimum spanning tree problem can be formalised in m-Kleene-algebras, and the following instance is the main target model for this.

In the weighted-graph model the operations $+$, s and m on finite matrices over extended real numbers can be defined as follows. First, consider the operation $+$ on $\mathbb{R}'$:

$+$	$\bot$	y	$\top$
$\bot$	$\bot$	y	$\top$
x	x	$x{+}y$	$\top$
$\top$	$\top$	$\top$	$\top$

This operation is lifted componentwise to yield the $+$ operation on matrices over $\mathbb{R}'$. To obtain the operation s, we apply the associative and commutative operation $+$ to all entries of a matrix. The result is stored in a fixed entry of the matrix and all other entries are set to $\bot$. Finally, for the operation m we find an entry of

```
1  input g
2  f ← ⊥                                          f = constructed forest, initially empty
3  h ← g                                          h = remaining edges to consider
4  while h ≠ ⊥ do                                 any remaining edges?
5     e ← m(h)                                    choose edge e with minimal weight
6     if e ≤ ¬(f^T* f*) then                      edge between different components?
7        f ← (f ⊓ ¬(⊤ e f^T*)) ⊔ (f ⊓ ⊤ e f^T*)^T ⊔ e   add edge to forest
8     h ← h ⊓ ē ⊓ ē^T                             remove edge from those to consider
9  end
10 output f
```

Figure 1: A relational version of Kruskal's minimum spanning tree algorithm

the matrix that is minimal but not $\bot$. If there are several such entries, m selects the first one using a fixed order of matrix entries. The resulting matrix has $\top$ at that entry and $\bot$ everywhere else. We moreover let $m(\bot) = \bot$. Finite matrices over $\mathbb{R}'$ form an m-Kleene-algebra with these operations. See [37] for a formal definition of these operations and [42] for a generalisation to other kinds of aggregation. We have extended the previous proofs to show that all instances discussed in these papers also satisfy the new axioms (16) and (17).

Axioms (17), (19) and (20) look more complex than the other axioms of m-Kleene-algebras. We conclude this section with a few remarks about why using these axioms in the correctness proof is not a problem in practice. Most of the correctness proof is carried out in Stone-Kleene relation algebras or more general structures; only a small part is concerned with the additional operations of m-algebras and their axioms. Simplicity of axioms does not necessarily translate to good automation; for example, the transitivity property $x \leq y \wedge y \leq z \Rightarrow x \leq z$ is arguably simple, yet some automated theorem provers struggle to find proofs involving chains of inequalities. Results requiring sizeable proofs are typically not proved directly from basic axioms of the underlying algebras; libraries of lemmas have to be developed using a combination of interactive and automated theorem proving. For example, axiom (20) is applied manually once in the proof, and each of its antecedents is dealt with separately. A benefit of working in m-Kleene-algebras instead of a particular concrete model is that all derived results hold in all instances of m-Kleene-algebras. In practice, this means that Kruskal's algorithm is correct for various optimisation problems based on different aggregation functions as discussed in [42].

7. Correctness of Kruskal's minimum spanning tree algorithm

In this section we present an algebraic version of Kruskal's minimum spanning tree algorithm [50] and prove its correctness. The algorithm is shown in Figure 1. It is a while-program with variables whose values range over an m-Kleene-algebra S.

The input of the algorithm is an undirected weighted graph $g \in S$. The algorithm constructs a minimum spanning forest $f \in S$ of the components of g and maintains the set of not-yet-considered edges h. The forest f is represented as a relation containing directed trees with roots; this is further explained below.

The algorithm starts with the empty forest f and all edges in h (lines 2 and 3). The while-loop is executed as long as there are further edges to consider (line 4). In each iteration an edge e is chosen with minimal weight among these edges (line 5). Note that e is an arc indicating a directed edge, so at the end of the iteration both e and its converse are removed from h (line 8). The forest is updated by the conditional, if e connects two different components of f (line 6). In this case, e is added to the forest and the remaining expression of the update makes sure that the components joined by e are again represented as a directed tree with a root (line 7). This update is further explained in Section 7.4. After all edges have been considered, the while-loop terminates and the output of the algorithm is f.

The result of the algorithm is an unweighted minimum spanning forest of g, where each component in the forest is a minimum spanning tree of a component of g. A weighted minimum spanning forest can be

obtained as the meet of the result and g. If g is connected, the result is a minimum spanning tree of g. But even if g is connected, Kruskal's algorithm has to work with forests, not just trees, as f may contain several components throughout the execution of the algorithm. This is a notable difference to Prim's algorithm, which maintains a tree throughout the construction.

We keep track of the constructed forest as a directed graph, in which each component is a directed tree with a root. This property is convenient to work with in Stone-Kleene relation algebras – by Definition 14 a forest is just an injective and acyclic element – but it has to be maintained throughout the algorithm. This is further detailed in Section 7.4. An example of a forest will be shown later in Figure 2.

We first discuss the tree components of a forest, which are used in the specification of the problem as well as in the correctness proof of Kruskal's algorithm.

7.1. Connected components of forests

For a relation x representing an undirected graph, the reflexive-transitive closure x^* is an equivalence relation representing the connected components of the graph. In particular, two nodes are related by x^* if and only if there is a path in x between the nodes, that is, if they belong to the same component. Recall that a relation is an equivalence if and only if it is reflexive, symmetric and transitive. The algebraic formulations of these properties are given in Definition 7.

If x represents a directed graph, $(x \sqcup x^\mathsf{T})^*$ gives the weakly connected components, which are obtained by ignoring the directions of edges. Moreover, $x^* \sqcap x^{\mathsf{T}*}$ gives the strongly connected components, again represented as an equivalence [63].

If x is a forest of directed trees with roots, the simplified expression $x^{\mathsf{T}*}x^*$ yields an equivalence representing the component trees. This is because, to connect two nodes, it suffices to have a path from the first node to the root of its component (backwards in x) followed by a path from this root to the second node (forwards in x). In fact, paths to any common ancestor of both nodes are sufficient. The following definition and subsequent result show properties of the components obtained in this situation.

Definition 20. *Let S be a Stone-Kleene relation algebra. The function $c : S \to S$ is defined by $c(x) = x^{\mathsf{T}*}x^*$.*

Theorem 21. *Let S be a Stone-Kleene relation algebra and let $x \in S$ be injective. Then*

1. *c is $\leq$-increasing and $\leq$-isotone*
2. *$c(x)$ is an equivalence*
3. *$c(x)^* = c(x)$*
4. *$c(c(x)) = c(x)$*

We use these properties for specifying spanning forests and for reasoning about the forest constructed by the algorithm.

7.2. Specification of the minimum spanning forest problem

The following definition gives formal specifications for the minimum spanning forest problem.

Definition 22. *Let S be an s-algebra and let $f, g \in S$. Then f is a* spanning forest *of g if f is a forest, f is regular and*

$$f \leq \overline{\overline{g}} \tag{22}$$

$$\overline{\overline{g}}^* \leq c(f) \tag{23}$$

Such an f is a minimum spanning forest *of g if, additionally,*

$$s(f \sqcap g) \leq s(u \sqcap g)$$

for each spanning forest u of g.

Inequality (22) states that all edges of f are contained in g (ignoring the weights), that is, that the forest f is a subgraph of g. Inequality (23) states that the components of g refine the components of f (both as equivalence relations). Note that $\overline{\overline{g}}$ ignores the weights of g and $\overline{\overline{g}}^*$ gives the components of this undirected graph, while $c(f) = f^{\mathsf{T}*} f^*$ gives the components for a forest of directed trees with roots.

An algorithm that solves the minimum spanning forest problem produces for any given finite undirected graph g a minimum spanning forest f of g. As usual, undirected means that g is symmetric.

7.3. Total-correctness proof

Our previous correctness proof of Prim's algorithm [37, 42] made two assumptions:

- The while-loop in the algorithm terminates.
- There exists a minimum spanning tree.

The corresponding assumption for Kruskal's algorithm is that there exists a minimum spanning forest. In the following we eliminate both the termination and the existence assumptions for Kruskal's algorithm; they can be eliminated similarly for Prim's algorithm.

A proof of termination relies on the fact that the graph is finite. This is formalised by axiom (17) of linear s-algebras.

Existence of a minimum spanning forest is usually proved by observing that there exists a spanning forest and that the number of spanning forests of a finite graph is finite. We use a termination proof to ensure that there exists a spanning forest, again axiom (17) to show that the number of spanning forests is finite and axiom (16) to show that one of them is a minimum spanning forest.

The assumption that a minimum spanning forest exists is essential to establish the invariant used in standard correctness proofs of spanning tree algorithms [23]. This assumption is separately discharged by the above observation. Termination of the algorithm (which can be proved independently) does not ensure that a *minimum* spanning forest exists as correctness of the algorithm relies on this assumption. However, termination of the algorithm can be used to prove that a spanning forest exists without relying on this assumption. This is exploited in the following approach:

1. Show that Kruskal's algorithm terminates and that its output is a spanning forest. This can be done without any further assumptions.
2. Conclude that a spanning forest exists. We effectively use the algorithm as a constructive existence proof. Any algorithm that constructs a spanning forest could be used for this step; using the same algorithm simplifies step 4 since parts of the proofs are identical.
3. Conclude that a minimum spanning forest exists.
4. Show that the output of Kruskal's algorithm is a minimum spanning forest. Use the fact obtained in step 3 to establish the invariant for this proof.

Steps 1 and 4 are correctness proofs in Hoare logic for the same program using the same precondition but different invariants and postconditions. Step 4 uses a stronger invariant and a stronger postcondition than step 1. The following four theorems give the precise statement for each step including the precondition, invariants, postconditions and bound function used in steps 1 and 4.

Theorem 23. *Let S be an m-Kleene-algebra and let $g \in S$ be symmetric. Then the following invariant holds throughout Kruskal's algorithm:*

- *g and h are symmetric*
- $g \sqcap \overline{\overline{h}} = h$
- *f is a spanning forest of $\overline{h} \sqcap g$*

Moreover, the following bound function is decreased by every iteration of the while-loop:

- $|\{x \in S \mid x = \overline{\overline{x}} \wedge x \leq \overline{\overline{h}}\}|$

Finally, the following postcondition is established:

- *f is a spanning forest of g*

The condition $g \sqcap \overline{\overline{h}} = h$ states that h is a weighted subgraph of g, that is, h contains a subset of the edges of g with the same weights as in g. Hence $\overline{h} \sqcap g$ contains those edges of g that have already been considered by the algorithm, and the third condition of the invariant states that f is a spanning forest of that part of g.

The bound function is a function depending on the state of the program; in this case, it only depends on the value of program variable h. It counts the number of regular elements (unweighted graphs) below $\overline{\overline{h}}$, which is the unweighted graph that contains all edges that have not been considered yet. Since h strictly decreases in each iteration, so does the number of regular elements below $\overline{\overline{h}}$. Note that we need to focus on the regular elements as there may be infinitely many non-regular elements (weighted graphs) even if only finite graphs are considered, since the number of different weights is not bounded.

Theorem 24. *Let S be an m-Kleene-algebra and let $g \in S$ be symmetric. Then there is an $f \in S$ such that f is a spanning forest of g.*

The preceding result follows from the definition of total-correctness Hoare triples. Specifically, for precondition p, statement c and postcondition q, the total-correctness triple $p\{c\}q$ means that for every program state s that satisfies p, there is a program state t satisfying q such that execution of c in starting state s ends in state t. The state transitions of c are given by an inductively defined operational semantics. This definition of Hoare triples uses the fact that the semantics is deterministic.

Theorem 25. *Let S be an m-Kleene-algebra and let $g \in S$ be symmetric. Then there is an $f \in S$ such that f is a minimum spanning forest of g.*

To show this, the following lemma is used, which holds in arbitrary partial orders. If X is a finite non-empty set whose image under a function v is linearly ordered, then X contains an element that minimises v. In our case, X is the set of spanning forests of g, which is finite by axiom (17) and non-empty by Theorem 24, and v is the operation s so that the image of X is linearly ordered by axiom (16).

Finally, we obtain a result similar to Theorem 23, but with an additional invariant and the desired, stronger postcondition.

Theorem 26. *Let S be an m-Kleene-algebra and let $g \in S$ be symmetric. Then the following invariant holds throughout Kruskal's algorithm:*

- *g and h are symmetric*
- *$g \sqcap \overline{\overline{h}} = h$*
- *f is a spanning forest of $\overline{h} \sqcap g$*
- *there is a minimum spanning forest w of g such that $f \leq w \sqcup w^{\mathsf{T}}$*

Moreover, the following bound function is decreased by every iteration of the while-loop:

- $|\{x \in S \mid x = \overline{\overline{x}} \wedge x \leq \overline{\overline{h}}\}|$

Finally, the following postcondition is established:

- *f is a minimum spanning forest of g*

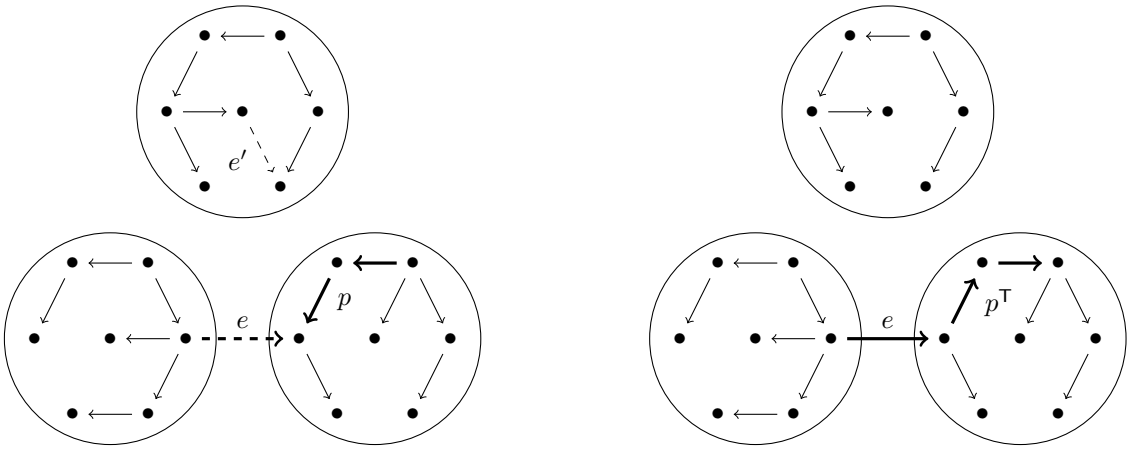

Figure 2: Forest f before update (left) and after update (right). Adding the edge e to the forest invalidates injectivity (left). Reversing the path p restores it (right). The edge e' is not added since it connects two nodes in the same component.

The additional invariant states that the currently constructed forest f can be extended to a minimum spanning forest w. Unlike for Prim's algorithm, this cannot be formalised by $f \leq w$ because different component trees of f may have their roots arranged such that adding another edge of g invalidates the tree structure. We therefore just require $f \leq w \sqcup w^\mathsf{T}$ so that the extension can use either direction of each edge of f.

All of these results have been proved in Isabelle/HOL based on a Hoare-logic library; see [54, 55]. To deal with total correctness we have modified the library and extended an existing tactic that automatically generates verification conditions for partial-correctness proofs along the lines suggested in [56] to total correctness. This requires every while-loop to be annotated with a bound function in addition to an invariant. See [61] for a comprehensive Hoare-logic framework.

In the above application, the generated conditions are predicates whose variables range over an m-Kleene-algebra; all calculations take place in this algebra or its reducts. The bound function maps elements of the m-Kleene-algebra to natural numbers.

7.4. Maintaining the Invariant

We discuss two parts of the correctness proof in more detail. The first part is how to maintain that f is a directed forest with roots in every iteration of the while-loop. The situation is illustrated in Figure 2 (left) showing three tree components of the forest f (which may contain further trees not given in the figure). The conditional in the while-loop of Kruskal's algorithm first checks whether the selected edge connects two nodes in the same component or in different components. If the edge connects nodes in the same component, such as e' in the figure, the forest is not changed.

A change only occurs for an edge e that connects different components of f. Simply adding e to f destroys the forest property as the target of e is not necessarily the root of its component tree. To restore the forest property, the path p backwards from the target of e to the root of its tree is reversed by forming

$$f' = (f \sqcap \overline{p}) \sqcup p^\mathsf{T}$$

as shown in Figure 2 (right). The required path is obtained by the expression

$$p = f \sqcap \top e f^{\mathsf{T}*},$$

which selects every edge of f whose target node is reachable from the target of e by going backwards in f. Other components of f are not affected. Hence, the new forest is $f' \sqcup e$, which (expanded and simplified) is the expression on the right-hand side of the assignment in line 7 of the algorithm in Figure 1.

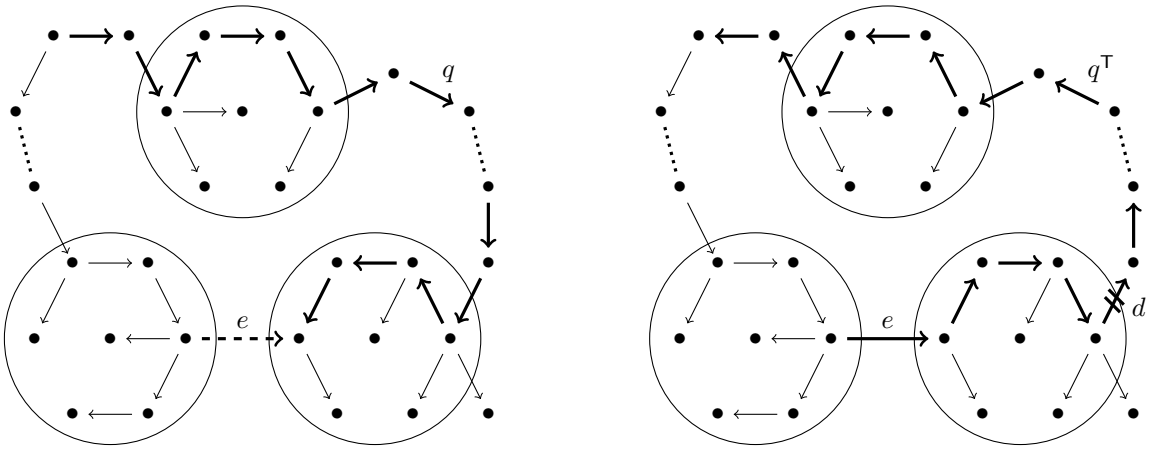

Figure 3: Minimum spanning forest w that extends f before update (left) and after update (right). Adding the edge e to the forest w invalidates injectivity (left). Reversing the path q restores it but creates a cycle (right). The cycle is broken by removing the edge d, which leaves the component of the target of the edge e.

The second part of the correctness proof we discuss is how to preserve the property that f can be extended to a minimum spanning forest. The situation is illustrated in Figure 3 (left) showing the same three components of the forest and further nodes. The displayed edges are those of a minimum spanning forest w that extends f as given in Figure 2 (left). Note that all edges of f occur in w if their direction is ignored; this is guaranteed by the invariant $f \leq w \sqcup w^\mathsf{T}$.

To maintain the invariant we need to construct a minimum spanning forest w' that extends f' as given in Figure 2 (right). Reversing the edges of p in w is not enough because the root of w does not have to be in the component of p, but can be anywhere in the graph. We proceed in two steps. In the first step we obtain the path q from the target of e to the root of w and reverse it by forming

$$v = (w \sqcap \overline{q}) \sqcup q^\mathsf{T}$$

as shown in Figure 3 (right). This is analogous to the construction of f' before and the required path is

$$q = w \sqcap \top e w^{\mathsf{T}*}.$$

Adding the edge e to v now creates a cycle. To restore the forest property we have to remove an edge from the cycle. The result still has to extend f' so the removed edge must not be e or contained in f or f^T. We choose the first edge d on the cycle that leaves the component of the target of e. This edge must exist because the cycle connects two different components of f, as the edge e shows. The required minimum spanning forest extending f' is thus obtained by

$$w' = (v \sqcap \overline{d}) \sqcup e.$$

The edge d is obtained by

$$d = v \sqcap \top e^\mathsf{T} v^{\mathsf{T}*} \sqcap c(f) e^\mathsf{T} \top \sqcap \top e \overline{c(f)},$$

where $c(f) = f^{\mathsf{T}*} f^*$ gives the components of f as described in Section 7.1. The various parts of this expression specify the following:

- $\top e^\mathsf{T} v^{\mathsf{T}*}$ requires that the target of d is reachable from the source of e by going backwards in v;
- $c(f) e^\mathsf{T} \top$ requires that the source of d is in the same component of f as the target of e;
- $\top e \overline{c(f)}$ requires that the target of d is not in the same component of f as the target of e.

It can be shown algebraically that d is an arc, that is, that d represents the unique edge satisfying these conditions. The first condition is required because there could be further edges in v leaving the target component of e, which are not on the cycle. The target of d becomes the root of w'.

Since the correctness proof is carried out in m-Kleene-algebras the result holds in numerous instances based on different aggregation operations given in [42].

See [37] for a discussion of related work on the verification of minimum spanning tree algorithms. In particular for Kruskal's algorithm we add that [29] describes a derivation in the B framework focussing on the underlying data structures, which are not a part of the present work. The B specification is based on sets and relations and weight functions thus involving objects of different sorts. According to [1], it is not clear whether all proof obligations have been discharged.

8. Conclusion

In the present paper we have studied algebras for modelling weighted graphs. Stone relation algebras are designed to stay so close to relation algebras that relational methods and concepts can be reused, yet be general enough to capture weighted graphs. Like relation algebras, Stone relation algebras can be combined with Kleene algebras for reasoning about reachability. All of our results about these algebraic structures have been formally verified in Isabelle/HOL; this includes a proof that weighted graphs represented by matrices over extended reals form an instance.

We have applied these results in three case studies. The first is a formally verified proof of Chen and Grätzer's construction theorem for Stone algebras [38]. It involves extensive reasoning about algebraic structures in addition to reasoning in algebraic structures. The second case study is a formal verification of Prim's minimum spanning tree algorithm [37]. Given in the present paper, the third case study formally verifies the correctness of Kruskal's algorithm for computing minimum spanning forests.

Both algorithm verifications use Hoare logic and most of the proofs can be carried out in Stone-Kleene relation algebras. The specification of the minimum spanning tree problem had to be modified to deal with Kruskal's algorithm. We expect that Prim's algorithm can be verified with respect to the new specification as trees are a special case of forests. Because both verifications are carried out in m-Kleene-algebras the proofs hold for a number of different instances as discussed in [42], which means that the algorithms solve a range of different problems.

Section 4 interprets a number of relational properties for weighted graphs. Future work will consider further graph algorithms to understand the limits of what can be expressed algebraically in this model. We expect that the total-correctness proof method described in Section 7.3 will be useful for other algorithms, too. The long-term goal of these efforts is a library for algebraic reasoning about weighted graphs and graph algorithms. This will require formalisation and theory development of domain-specific concepts at a higher level than the relation-algebraic formulas used, for example, in Section 7.4.

Acknowledgements

I thank the anonymous referees for their helpful feedback. Using $s(z)$ instead of z in axiom (12) has been suggested by one of the anonymous referees.

References

[1] J.-R. Abrial, D. Cansell, and D. Méry. Formal derivation of spanning trees algorithms. In D. Bert, J. P. Bowen, S. King, and M. Waldén, editors, *ZB 2003: Formal Specification and Development in Z and B*, volume 2651 of *LNCS*, pages 457–476. Springer, 2003.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley Publishing Company, 1974.

[3] H. Andréka and Sz. Mikulás. Axiomatizability of positive algebras of binary relations. *Algebra Universalis*, 66(1–2):7–34, 2011.

[4] A. Armstrong, S. Foster, G. Struth, and T. Weber. Relation algebra. *Archive of Formal Proofs*, 2016, first version 2014.

[5] A. Armstrong, V. B. F. Gomes, G. Struth, and T. Weber. Kleene algebra. *Archive of Formal Proofs*, 2016, first version 2013.

[6] T. Asplund. Formalizing the Kleene star for square matrices. Bachelor Thesis IT 14 002, Uppsala Universitet, Department of Information Technology, 2014.
[7] R. C. Backhouse and B. A. Carré. Regular algebra applied to path-finding problems. *Journal of the Institute of Mathematics and its Applications*, 15(2):161–186, 1975.
[8] R. Balbes and P. Dwinger. *Distributive Lattices*. University of Missouri Press, 1974.
[9] R. Balbes and G. Grätzer. Injective and projective Stone algebras. *Duke Mathematical Journal*, 38(2):339–347, 1971.
[10] 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.
[11] R. Berghammer, A. Rusinowska, and H. de Swart. Computing tournament solutions using relation algebra and RelView. *European Journal of Operational Research*, 226(3):636–645, 2013.
[12] R. Berghammer and B. von Karger. Relational semantics of functional programs. In C. Brink, W. Kahl, and G. Schmidt, editors, *Relational Methods in Computer Science*, chapter 8, pages 115–130. Springer, Wien, 1997.
[13] R. Berghammer, B. von Karger, and A. Wolf. Relation-algebraic derivation of spanning tree algorithms. In J. Jeuring, editor, *MPC 1998*, volume 1422 of *LNCS*, pages 23–43. Springer, 1998.
[14] R. Bird and O. de Moor. *Algebra of Programming*. Prentice Hall, 1997.
[15] G. Birkhoff. *Lattice Theory*, volume XXV of *Colloquium Publications*. American Mathematical Society, third edition, 1967.
[16] S. Bistarelli and F. Santini. C-semiring frameworks for minimum spanning tree problems. In A. Corradini and U. Montanari, editors, *Recent Trends in Algebraic Development Techniques*, volume 5486 of *LNCS*, pages 56–70. Springer, 2009.
[17] J. C. Blanchette, S. Böhme, and L. C. Paulson. Extending Sledgehammer with SMT solvers. In N. Bjørner and V. Sofronie-Stokkermans, editors, *CADE 2011*, volume 6803 of *LNCS*, pages 116–130. Springer, 2011.
[18] J. C. Blanchette and T. Nipkow. Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In M. Kaufmann and L. C. Paulson, editors, *ITP 2010*, volume 6172 of *LNCS*, pages 131–146. Springer, 2010.
[19] T. S. Blyth. *Lattices and Ordered Algebraic Structures*. Springer, 2005.
[20] D. A. Bredihin and B. M. Schein. Representations of ordered semigroups and lattices by binary relations. *Colloquium Mathematicum*, 39(1):1–12, 1978.
[21] S. D. Comer. On connections between information systems, rough sets and algebraic logic. In C. Rauszer, editor, *Algebraic Methods in Logic and in Computer Science*, volume 28 of *Banach Center Publications*, pages 117–124. Institute of Mathematics, Polish Academy of Sciences, 1993.
[22] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.
[23] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
[24] H. B. Curry. *Foundations of Mathematical Logic*. Dover Publications, 1977.
[25] J. Desharnais, A. Grinenko, and B. Möller. Relational style laws and constructs of linear algebra. *Journal of Logical and Algebraic Methods in Programming*, 83(2):154–168, 2014.
[26] J. Desharnais and G. Struth. Internal axioms for domain semirings. *Science of Computer Programming*, 76(3):181–203, 2011.
[27] I. Düntsch, G. Gediga, and E. Orłowska. Relational attribute systems II: Reasoning with relations in information structures. In J. F. Peters, A. Skowron, V. W. Marek, E. Orłowska, R. Słowiński, and W. Ziarko, editors, *Transactions on Rough Sets VII*, volume 4400 of *LNCS*, pages 16–35. Springer, 2007.
[28] I. Düntsch and M. Winter. Rough relation algebras revisited. *Fundamenta Informaticae*, 74(2–3):283–300, 2006.
[29] R. Fraer. Minimum spanning tree. In E. Sekerinski and K. Sere, editors, *Program Development by Refinement*, chapter 3, pages 79–114. Springer, 1999.
[30] P. J. Freyd and A. Ščedrov. *Categories, Allegories*, volume 39 of *North-Holland Mathematical Library*. Elsevier Science Publishers, 1990.
[31] M. F. Frias, N. Aguayo, and B. Novak. Development of graph algorithms with fork algebras. In *XIX Conferencia Latinoamericana de Informática*, pages 529–554, 1993.
[32] E. Fried, G. E. Hansoul, E. T. Schmidt, and J. C. Varlet. Perfect distributive lattices. In G. Eigenthaler, H. K. Kaiser, W. B. Müller, and W. Nöbauer, editors, *Contributions to General Algebra 3*, pages 125–142. Hölder-Pichler-Tempsky, 1985.
[33] J. A. Goguen. L-fuzzy sets. *Journal of Mathematical Analysis and Applications*, 18(1):145–174, 1967.
[34] M. Gondran and M. Minoux. *Graphs, Dioids and Semirings*. Springer, 2008.
[35] G. Grätzer. *Lattice Theory: First Concepts and Distributive Lattices*. W. H. Freeman and Co., 1971.
[36] W. Guttmann. Algebras for iteration and infinite computations. *Acta Informatica*, 49(5):343–359, 2012.
[37] W. Guttmann. Relation-algebraic verification of Prim's minimum spanning tree algorithm. In A. Sampaio and F. Wang, editors, *ICTAC 2016*, volume 9965 of *LNCS*, pages 51–68. Springer, 2016.
[38] W. Guttmann. Stone algebras. *Archive of Formal Proofs*, 2016.
[39] W. Guttmann. Stone-Kleene relation algebras. *Archive of Formal Proofs*, 2017.
[40] W. Guttmann. Stone relation algebras. In P. Höfner, D. Pous, and G. Struth, editors, *Relational and Algebraic Methods in Computer Science*, volume 10226 of *LNCS*, pages 127–143. Springer, 2017.
[41] W. Guttmann. Stone relation algebras. *Archive of Formal Proofs*, 2017.
[42] W. Guttmann. An algebraic framework for minimum spanning tree problems. *Theoretical Computer Science*, 2018. https://doi.org/10.1016/j.tcs.2018.04.012.
[43] R. Hirsch and I. Hodkinson. *Relation Algebras by Games*. Elsevier Science B.V., 2002.
[44] P. Höfner and B. Möller. Dijkstra, Floyd and Warshall meet Kleene. *Formal Aspects of Computing*, 24(4):459–476, 2012.
[45] Y. Kawahara and H. Furusawa. Crispness in Dedekind categories. *Bulletin of Informatics and Cybernetics*, 33(1–2):1–18,

2001.
[46] Y. Kawahara, H. Furusawa, and M. Mori. Categorical representation theorems of fuzzy relations. *Information Sciences*, 119(3–4):235–251, 1999.
[47] D. Killingbeck, M. S. Teixeira, and M. Winter. Relations among matrices over a semiring. In W. Kahl, M. Winter, and J. N. Oliveira, editors, *RAMiCS 2015*, volume 9348 of *LNCS*, pages 101–118. Springer, 2015.
[48] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.
[49] D. Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems*, 19(3):427–443, 1997.
[50] J. B. Kruskal, Jr. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.
[51] H. D. Macedo and J. N. Oliveira. A linear algebra approach to OLAP. *Formal Aspects of Computing*, 27(2):283–307, 2015.
[52] R. D. Maddux. Relation-algebraic semantics. *Theoretical Computer Science*, 160(1–2):1–85, 1996.
[53] R. D. Maddux. *Relation Algebras*. Elsevier B.V., 2006.
[54] T. Nipkow. Winskel is (almost) right: Towards a mechanized semantics textbook. *Formal Aspects of Computing*, 10(2):171–186, 1998.
[55] T. Nipkow. Hoare logics in Isabelle/HOL. In H. Schwichtenberg and R. Steinbrüggen, editors, *Proof and System-Reliability*, pages 341–367. Kluwer Academic Publishers, 2002.
[56] T. Nipkow and G. Klein. *Concrete Semantics*. Springer, 2014.
[57] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.
[58] E. Orłowska and A. M. Radzikowska. Double residuated lattices and their applications. In H. C. M. de Swart, editor, *Relational Methods in Computer Science*, volume 2561 of *LNCS*, pages 171–189. Springer, 2002.
[59] L. C. Paulson and J. C. Blanchette. Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In G. Sutcliffe, E. Ternovska, and S. Schulz, editors, *Proceedings of the 8th International Workshop on the Implementation of Logics*, pages 3–13, 2010.
[60] Z. Pawlak. Rough sets, rough relations and rough functions. *Fundamenta Informaticae*, 27(2–3):103–108, 1996.
[61] N. Schirmer. *Verification of Sequential Imperative Programs in Isabelle/HOL*. PhD thesis, TU München, 2006.
[62] J. Schmid. Algebraically closed distributive p-algebras. *Algebra Universalis*, 15:126–141, 1982.
[63] G. Schmidt and T. Ströhlein. *Relationen und Graphen*. Springer, 1989.
[64] G. Struth. Abstract abstract reduction. *Journal of Logic and Algebraic Programming*, 66(2):239–270, 2006.
[65] A. Tarski. On the calculus of relations. *The Journal of Symbolic Logic*, 6(3):73–89, 1941.
[66] M. Winter. A new algebraic approach to L-fuzzy relations convenient to study crispness. *Information Sciences*, 139(3–4):233–252, 2001.