Dijkstra, Floyd and Warshall meet Kleene

Peter Höfner, Bernhard Möller

Angaben zur Veröffentlichung / Publication details:

Höfner, Peter, and Bernhard Möller. 2012. "Dijkstra, Floyd and Warshall meet Kleene." *Formal Aspects of Computing* 24 (4-6): 459–76. https://doi.org/10.1007/s00165-012-0245-4.

Under consideration for publication in Formal Aspects of Computing

Dijkstra, Floyd and Warshall Meet Kleene

Peter Höfner[1,2] and Bernhard Möller[3]

[1]NICTA, Australia
[2]University of New South Wales, Australia
[3]Universität Augsburg, Germany

It is our pleasure to dedicate this paper to Charles Carroll Morgan at the occasion on his 60th birthday. With this paper we try to combine two of Carroll's various research interests: Programming Methodology and Algorithmic Calculi. The former includes problem-solving techniques, such as algorithms, the latter includes the study of calculation of programs from specifications, in particular, formal refinement techniques. We illustrate this with a fresh look at path problems in graphs, a topic which is regaining interest in connection with protocol verification, another of Carroll's interests.

Abstract. Around 1960, Dijkstra, Floyd and Warshall published papers on algorithms for solving single-source and all-sources shortest path problems, respectively. These algorithms, nowadays named after their inventors, are well known and well established. This paper sheds an algebraic light on these algorithms. We combine the shortest path problems with Kleene algebra, also known as Conway's regular algebra. This view yields a purely algebraic version of Dijkstra's shortest path algorithm and the one by Floyd/Warshall. Moreover, the algebraic abstraction yields applications of these algorithms to structures different from graphs and pinpoints the mathematical requirements on the underlying cost algebra that ensure their correctness.

1. Introduction

> "We consider n points (nodes), some or all pairs of which are connected by a branch; the length of each branch is given. We restrict ourselves to the case where at least one path exists between any two nodes."
>
> "**Problem 1.** Construct the tree of minimum total length between the n nodes."
> "**Problem 2.** Find the path of minimum total length between two given nodes P and Q."
>
> E. W. Dijkstra [Dij59]

The solution of the problem given by Dijkstra is nowadays known as *Dijkstra's shortest path algorithm.* It solves a single-source shortest path problem for a (finite) graph with non-negative edge costs by iteration, producing a shortest-path tree. The original algorithm keeps a set of approximations to shortest paths from a source s to the other nodes, improving those approximations in each iteration of the algorithm. To do so, it defines two sets of nodes, one for which the shortest path problem has already been solved and one which contains all the other nodes.

Correspondence and offprint requests to: Bernhard Möller, Universität Augsburg, Universitätsstr. 6a, 86135 Augsburg, Germany. e-mail: bernhard.moeller@informatik.uni-augsburg.de

In his seminal paper [Dij59], Dijkstra presents the algorithm in a non-formal way[1], but does not give a correctness proof, this being done later e.g. in [AHU74]. Nevertheless he already states the essential idea of the algorithm's invariant in an informal way.

In this paper we give a purely algebraic derivation for Dijkstra's shortest path algorithm. This immediately shows its correctness and the relationship to the algorithm of Floyd/Warshall, also known as Floyd's algorithm or Roy/Warshall algorithm [Roy59, Flo62, War62]. A similar algebraic treatment in a more concrete setting of matrices, based on ideas of [BC75], has been presented in [BEG94]. Our algebraic derivation abstracts further and extends the algorithm to an entire class of mathematical structures known as *Kleene algebras* [Con71, Koz94]. These structures are *idempotent semirings* (e.g. [GM08]), enriched by the iteration operation of Kleene star [Kle51]. An essential ingredient for motivating the abstract treatment is the fact that $n \times n$-matrices over Kleene algebras again form a Kleene algebra [Con71]. As a further refinement, we use Kleene algebras with *tests* [Koz97], which allow a more compact and purely first-order algebraic treatment of sets of nodes in a graph.

Since the derivation is independent of a particular underlying algebra, we may choose one that is particularly suited to the application to routing protocols we are aiming at. To this end, as another contribution of this paper, we define a new algebra of path sets. It allows keeping track of all minimum-cost paths from the source s to a given target x and hence, in particular, of the set of all immediate predecessors of x on these paths.

The paper is organised as follows. In Section 2, the shortest-path problem is characterised in an algebraic fashion. To achieve this we also present a new algebra working with path sets. In the following Section 3, we abstract from the concrete specification of the problem and present a generic algebraic specification together with all preliminaries needed for the derivation of the algorithm later on. In Section 4, we present a stepwise refinement starting from the specification given before, which finally yields Dijkstra's algorithm. A similar algebraic derivation that leads to the algorithm of Floyd/Warshall is presented in Section 5. In Section 6 further applications of the algebraic approach are presented: we discuss the consequences of the algorithms presented when the underlying algebraic cost model is changed. The examples chosen are particularly useful for routing problems in networks. The paper is rounded off by a discussion on related work (Section 7) and a short conclusion (Section 8).

2. Shortest Paths Algebraically

Before we present a derivation of Dijkstra's algorithm, we have to formulate the problem in an algebraic manner. Of course the problem is based on paths.

A *path* over a set Σ of nodes is a non-empty and finite sequence $p = v_0.v_1 \dots v_n$ of $n+1$ nodes $v_i \in \Sigma$ for some $n \in \mathbb{N}$, i.e., an element of Σ^+. The subpath $v_i \dots v_j$ of a path p for $0 \le i \le j \le n$ is denoted by $p_{[i..j]}$. A path with exactly two nodes is called an *edge*. In the sequel we use three special subpaths: $\mathit{first}(p) =_{df} p_{[0..0]}$ is the *first* element (*source*) of a path p; $\mathit{tail}(p) =_{df} p_{[1..n]}$ is the *tail* of a path p with $n+1$ nodes; and $\mathit{last}(p) =_{df} p_{[n..n]}$ denotes the *last* element of a path p consisting of $n+1$ nodes.

For simplicity, we take the view that each edge in a path has unit cost; we will refine this in Section 6.1. The cost of a path $p = v_0.v_1 \dots v_n$ is given by its *length* n, i.e., by its number of edges $v_i.v_{i+1}$, which we denote by $|p|$. Note that p is not the cardinality of the set of these edges; multiple occurrences are counted separately. For example, $|v_1.v_2.v_3| = 3 \neq 5 = |v_1.v_2.v_1.v_2.v_3|$. Concatenation or path fusion glues two paths together if the last node of the left path equals the first node of the right path, and removes the duplicate:

$$p \bowtie q =_{df} \begin{cases} p \,.\, \mathit{tail}(q) & \text{if } \mathit{last}(p) = \mathit{first}(q) \\ \text{undefined} & \text{otherwise .} \end{cases} \tag{1}$$

Given a set P of paths and nodes $v, w \in \Sigma$, we set

$$\mathit{paths}(v, w, P) =_{df} \{p \in P \mid \mathit{first}(p) = v \wedge \mathit{last}(p) = w\} \ ,$$

which selects all those paths of P that start in v and end in w.

This enables a matrix representation of path sets, see for instance [GM08]:

[1] By this we mean that the algorithm is carefully described using English text, but without formal methods and techniques as one would do today.

Proposition 2.1. *For a finite set Σ with $|\Sigma| = n$, every set P of paths is isomorphic to an $n \times n$-matrix M with the nodes in Σ as row and column indices and subsets of P as entries, namely $M_{vw} = paths(v, w, P)$ for $v, w \in \Sigma$.*

A path set P is *uniform* if all paths in it that connect a fixed pair of nodes have the same length, i.e.,

$$\forall\, v, w \in \Sigma : \forall\, p, q \in paths(v, w, P) : |p| = |q| \ .$$

We note that $\emptyset$ and Σ (considered as sets of single-node paths) are uniform. The set of all uniform path sets over Σ is denoted by $\mathrm{UP}(\Sigma)$. The set $\mathrm{UP}(\Sigma)$ is closed under forming subsets.

We define the *distance* between two nodes v, w according to a set P of paths between v, w as the length of a shortest path in P:

$$dist(v, w, P) =_{df} \texttt{inf}\,\{|p| \,|\, p \in paths(v, w, P)\} \ , \tag{2}$$

with $\texttt{inf}\,\emptyset =_{df} \infty$. This latter convention reflects that absence of a path between u and v represents an unsurmountable obstacle of infinite cost.

The following operator selects the subset of paths with shortest length from given path set P:

$$shortest(P) =_{df} \{p \,|\, p \in P \wedge \forall\, q \in P : |p| \leq |q|\} \ .$$

By definition, $shortest(P)$ is uniform. We combine this with the operator that determines the connecting paths between pairs of nodes:

$$shortpaths(v, w, P) =_{df} shortest(paths(v, w, P)) \ .$$

Choice compares two paths with the same source and the same target. If possible, it chooses the shorter of the two; in case the lengths are the same, it keeps both paths. In lifting this to sets we have to group the paths involved into subsets having equal source and target nodes:

$$P + Q =_{df} \bigcup_{v,w \in \Sigma} paths(v, w, P) \oplus paths(v, w, Q) \ ,$$

where

$$paths(v, w, P) \oplus paths(v, w, Q) =_{df} \begin{cases} paths(v, w, P) & \text{if } dist(v, w, P) < dist(v, w, Q) \\ paths(v, w, Q) & \text{if } dist(v, w, Q) < dist(v, w, P) \\ paths(v, w, P) \cup paths(v, w, Q) & \text{if } dist(v, w, Q) = dist(v, w, P) \ . \end{cases}$$

If $paths(v, w, P)$ or $paths(v, w, Q)$ is empty, the respective distance is infinite and one chooses the other set. Choice $+$ is associative and preserves uniformity. Moreover, by definition, it is commutative and has $\emptyset$ as its unit.

To calculate at the same algebraic level, we also lift path fusion to uniform sets of paths:

$$P \cdot Q =_{df} \bigcup_{u,v,w \in \Sigma} shortest(paths(u, v, P) \bowtie paths(v, w, Q)) \ ,$$

where $\bowtie$ is lifted pointwise to set of paths, i.e., $P \bowtie Q = \{p \bowtie q \,|\, p \in P \wedge q \in Q \wedge p \bowtie q \text{ is defined}\}$. This arises from gluing paths from u to v and from v to w to paths from u to w and keeping only the shortest of these.

It is easy to verify that $\cdot$, like $+$, is associative and preserves uniformity. Its neutral element is Σ. Therefore it makes sense to define the powers of a path set P, i.e., its repeated fusion with itself, by

$$P^0 = \Sigma \qquad \text{and} \qquad P^{i+1} = P \cdot P^i$$

and its arbitrary finite iteration by

$$P^* =_{df} \sum_{i \in \mathbb{N}} P^i \ . \tag{3}$$

A *graph* over Σ consists of a set $E \subseteq \Sigma^2$ of edges. A particular instance is the *fully connected* graph Σ^2. For a graph E and a source node $s \in \Sigma$, the specification of the single-source shortest path problem then reads

$$\textbf{Calculate } \{s\} \cdot E^* \ . \tag{4}$$

To derive an efficient algorithm which solves the single-source shortest path problem, one has to eliminate the expensive star operation. In Section 4, such an elimination will be performed by a stepwise refinement in an abstract algebraic setting, yielding Dijkstra's algorithm. Before that, we present the algebraic background needed.

3. Algebraic Background—Semirings and Kleene Algebras

In this section, we recapitulate all theory needed for the algebraic treatment of shortest-path algorithms. In particular, semirings and Kleene algebras are defined and some basic properties are discussed. Last, we show the relationship between set of (uniform) paths, as introduced before, and the algebra. We will return to the algebraic derivation of shortest-paths algorithms in Section 4 ff.

Proposition 3.1. *For uniform path sets $P, Q, R \in \mathrm{UP}(\Sigma)$, concatenation ($\cdot$) distributes over choice ($+$).*

$$P \cdot (Q + R) = P \cdot Q + P \cdot R \ ,$$
$$(Q + R) \cdot P = Q \cdot P + R \cdot P \ ,$$

Moreover, choice is idempotent: $P + P = P$.

We now define the corresponding abstract algebraic framework, where we use arbitrary elements instead of just sets of paths.

A *monoid* is a triple $(S, \circ, e)$ where S is a set, $\circ$ is a binary operator on S and $e \in S$ is an element such that $\circ$ is associative and e is the unit, i.e., the neutral element of $\circ$. The monoid is called *commutative* if $\circ$ is a commutative operation. It is called *idempotent* if $\circ$ satisfies $a \circ a = a$ for all $a \in S$.

An *idempotent semiring (i-semiring)* or *dioid* is a quintuple $(S, +, \cdot, \mathbf{0}, \mathbf{1})$ such that $(S, +, \mathbf{0})$ is an idempotent and commutative monoid, $(S, \cdot, \mathbf{1})$ is a monoid, multiplication distributes over addition from the left and right and $\mathbf{0}$ is a left and right zero of multiplication, i.e., $\mathbf{0} \cdot a = a \cdot \mathbf{0} = \mathbf{0}$. We use the convention that composition binds stronger than choice.

Our properties entail the following result.

Proposition 3.2. *The structure $(\mathrm{UP}(\Sigma), +, \cdot, \emptyset, \Sigma)$ forms an i-semiring.*

Every i-semiring comes with a partial order $\leq$, called *natural order* and defined by $a \leq b \Leftrightarrow_{df} a + b = b$. The operations $+$ and $\cdot$ are *isotone*, i.e., monotonically increasing in both arguments w.r.t. $\leq$. In particular, $\mathbf{0}$ is the least element. Moreover, equations can be split into two inequations, i.e., $a = b \Leftrightarrow a \leq b \wedge b \leq a$. The order also induces an upper semilattice in which $a + b$ is the supremum of a and b, and $\mathbf{0}$ is the least element. This means it is characterised by the universal property

$$a + b \leq c \Leftrightarrow a \leq c \wedge b \leq c \ . \tag{5}$$

For the i-semiring of uniform path sets the relation $P \leq Q$ means that Q offers, for all pairs of sources and targets, less costly (shorter) paths than P. The empty set $\emptyset$, which offers no paths at all, therefore is the worst set.

Many i-semirings have a greatest element $\top$. If it exists it is characterised by $a \leq \top$. In the semiring of uniform path sets, $\top = \Sigma^2 + \Sigma$, since $\top$ lists all shortest (optimal) paths between all pairs of nodes: the shortest path between two different nodes v_1 and v_2 is a direct link of length one—an element from Σ^2; the shortest path between v_1 and v_1 is "stepping on the spot"—an element of Σ.

Next we look at the algebraic properties of subsets of Σ, i.e., sets of nodes considered as sets of single-node paths. By definition they are uniform. If $P \subseteq \Sigma$ is such a set then, for arbitrary set Q of paths, the set $P \cdot Q$ consists, by the definition of path gluing, of all those paths in Q that start with an element of P. In other words, forming $P \cdot Q$ restricts Q to the starting set P. Symmetrically, $Q \cdot P$ consists of all those paths in Q that end with an element of P. Both restrictions are subsets of Q; this means that by restricting one may lose some cheap paths and hence obtain a worse, i.e., more costly, path set. We will restate this in more general terms below. If Q is also a subset of Σ then $P \cdot Q = Q \cdot P = P \cap Q$. If Q consists exactly of those single-node paths that are not in P, i.e., $Q = \overline{P} \cap \Sigma$, then $P + Q = \Sigma$ and $P \cdot Q = \emptyset$.

Here is the abstract algebraic counterpart (remember that in the i-semiring of path sets Σ plays the rôle of $\mathbf{1}$, the unit of $\cdot$). An element $p \leq \mathbf{1}$ of an i-semiring S is called a *test* if it has a relative complement,

denoted $\neg p$ satisfying

$$p + \neg p = \mathbf{1} \ , \qquad p \cdot \neg p = \mathbf{0} \ .$$

We will denote general i-semiring elements by $a, b, c, \ldots$ and tests by $p, q, r, \ldots$ (p for "predicate").

It can be shown that the above definition characterises $\neg p$ uniquely. The elements $\mathbf{0}$ and $\mathbf{1}$ are tests that are relative complements of each other. If $\mathbf{0}$ and $\mathbf{1}$ are the only tests, the i-semiring is called *test-discrete*. Moreover, the set $\mathtt{test}(S)$ of all tests of S forms a Boolean algebra in which $+$ coincides with join (or logical disjunction of predicates) and $\cdot$ coincides with meet (or logical conjunction). In particular, $\cdot$ is idempotent and commutative on tests. Two tests p, q are *disjoint*, if $p \cdot q = 0$. By definition we have p, q are disjoint $\Leftrightarrow\ p \leq \neg q \ \Leftrightarrow\ q \leq \neg p$.

Given a test p and an arbitrary i-semiring element a, we call $p \cdot a$ the *source restriction* and $a \cdot p$ the *target restriction* of a by p. By $p \leq \mathbf{1}$ and isotony of $\cdot$ one obtains $p \cdot a \leq a$ and $a \cdot p \leq a$. Moreover, if two elements have disjoint sources or targets, splitting rules for equational reasoning can be derived: if p, q are disjoint, then

$$\begin{array}{lcl} a \cdot p + b \cdot q = c \cdot p + d \cdot q & \Leftrightarrow & a \cdot p = c \cdot p \ \wedge\ b \cdot q = d \cdot q \ , \text{ and} \\ p \cdot a + q \cdot b = p \cdot c + q \cdot d & \Leftrightarrow & p \cdot a = p \cdot c \ \wedge\ q \cdot b = q \cdot d \ . \end{array} \tag{6}$$

A proof is given in Appendix A.

A *point* in an i-semiring is a test $p \neq \mathbf{0}$ that is atomic in $\mathtt{test}(S)$, i.e., has no proper subtest other than $\mathbf{0}$:

$$\forall\, q \in \mathtt{test}(S) : q \leq p \ \Rightarrow\ q = \mathbf{0} \ \vee\ q = p \ .$$

The reverse implication holds for all elements p, q of an arbitrary i-semiring. Therefore for a point q we can use an equivalence rather than an implication. In the i-semiring of uniform path sets, a point is a singleton subset of Σ, i.e., a set with a single path having just a single node. Since such a singleton single-node path with node x is always the cheapest path from x to itself (cost 0), we have the following fundamental property which will be central for our derivation:

$$\forall\, p, a \in S : p \text{ a point } \Rightarrow\ p \cdot a \cdot p \ \leq\ p \ . \tag{no detours}$$

An i-semiring S satisfying this property is called *a cost semiring*. When graphs and paths are considered, the no detours rule forbids cycles with negative weights, i.e., adding a cycle always increases the costs of a path. In a test-discrete i-semiring, the only point is $\mathbf{1}$ and therefore the no-detours rule is equivalent to

$$\forall\, a \in S : a \leq \mathbf{1} \ . \tag{7}$$

Now we enrich i-semirings by the concept of iteration. A *Kleene algebra* is a tuple $(S, +, \cdot, \mathbf{0}, \mathbf{1}, {}^*)$ such that $(S, +, \cdot, \mathbf{0}, \mathbf{1})$ is an i-semiring and the unary operation * satisfies the following axioms.

$$\mathbf{1} + a \cdot a^* = a^* \ , \qquad \mathbf{1} + a^* \cdot a = a^* \ , \tag{star unfold}$$

$$b + a \cdot c \leq c \Rightarrow a^* \cdot b \leq c \ , \qquad b + c \cdot a \leq c \Rightarrow b \cdot a^* \leq c \ . \tag{star induction}$$

The above axioms uniquely determine * when it exists. One of the two unfold laws could be dropped from the set of axioms, since it can be derived from the other axioms (e.g. [Hol98]). Moreover, they may be given as inequations since $a^* \leq \mathbf{1} + a \cdot a^*$ and $a^* \leq \mathbf{1} + a^* \cdot a$ can be immediately derived by star induction. Other laws that immediately follow from the axioms are

$$a^* \cdot a^* \ =\ a^* \ , \quad a \cdot a^* = a^* \cdot a \leq a^* \ , \quad a \leq a^* \ , \quad \text{and } \mathbf{1} \leq a^* \ . \tag{KA}$$

In the i-semiring of path sets the operation * as defined in (3) indeed satisfies these axioms.

Lemma 3.3. *The i-semiring of uniform path sets enriched by * forms a Kleene algebra.*

The lemma follows from Kleene's fixed point theorem [Kle52, DP02], since we are working in a power set lattice, which is complete, and the recursive functions associated with the star unfold laws are continuous in that lattice.

We state two properties that will be most useful in our derivation of Dijkstra's algorithm.

$$(a + b)^* \ =\ a^* \cdot (b \cdot a^*)^* \ =\ (a^* \cdot b) \cdot a^* \ , \tag{star of sum}$$

$$(a + b)^* \ =\ b^* \cdot (\mathbf{1} \ +\ a \cdot (b + a)^*) \ =\ (\mathbf{1} \ +\ (a + b)^* \cdot a) \cdot b^* \ . \tag{grouping}$$

The proofs can be found in Appendix A. The first line describes how an iteration of a sum of two elements can be rephrased using multiplication only. It roughly says that if one has always the choice between a and b, one takes a until b is chosen, after that a is chosen again for some time (maybe a is not chosen at all). The latter property means that an iterated arbitrary alternation between a and b can always be regrouped to exhibit a maximally long b-sequence at its beginning or end.

Finally, we note that all the above axioms are purely stated in first-order logic. This allows the use of automated first-order theorem provers for validating and verifying properties. It has been shown that the algebraic structures used are particularly amenable to automated reasoning [HS07].

If elements of semirings/Kleene algebras characterise single transitions, whole transition systems can be encoded in matrices. To calculate with matrices we recapitulate a well known result.

Theorem 3.4. *Standard operations of matrix addition and multiplication turn the family $M(n, K)$ of $n\times n$-matrices over a Kleene algebra K into a Kleene algebra again; the zero matrix is neutral w.r.t. $+$, and the identity matrix I is neutral w.r.t. multiplication.*

This result will be used in Section 4. A proof can e.g. be found in [Con71].

In the matrix model, the test set consists of all diagonal matrices with tests of the underlying i-semiring on their diagonals. A test is a point iff it has exactly one point of the underlying i-semiring on its diagonal and $\mathbf{0}$ everywhere else. If p, q are points, the nodes of which have indices i, j, and a is a matrix then $b =_{df} p\cdot a\cdot q$ is a matrix with entry $b_{ij} = p_{ii}\cdot a_{ij}\cdot q_{jj}$ and $\mathbf{0}$ everywhere else. In particular, setting $q = p$ shows that the matrix i-semiring is a cost semiring when the underlying i-semiring is.

4. Dijkstra's Algorithm

We now derive Dijkstra's algorithm. Abstracting the specification (4), the goal is to compute, for an arbitrary i-semiring S with $s \in S$ being a point and $e \in S$ an arbitrary element,

$$d =_{df} s\cdot e^* \,, \tag{8}$$

without using the expensive transitive-closure operation.

A standard technique for obtaining an iterative solution for a problem is to make constants in the problem specification into parameters; these additional degrees of freedom can then often be used to obtain an inductive solution. Sometimes the constants in question are "invisible"; for instance, they may be neutral elements w.r.t. some binary operator and have to be made explicit before the derivation proper can start. In (8) we can exhibit the neutral element $\mathbf{1}$ next to the set e of edges under the star:

$$d = s\cdot e^* = s\cdot(\mathbf{1}\cdot e)^* \,.$$

If we now replace this occurrence by a parameter we can control the set of edges that are considered by specifying the nodes that may be used as inner nodes in the paths analysed. To this end we define a function dd that solves the original problem for a restricted graph; we model the restricting set of nodes again as a test called ok[2]:

$$dd(ok) =_{df} s\cdot(ok\cdot e)^* \,.$$

$dd(ok)$ only considers paths whose intermediate nodes lie in ok, while their target nodes might be outside ok. Using isotony of * it is easy to see that dd is indeed an approximation of the shortest path problem, i.e.,

$$\forall\, ok : dd(ok) \le d \,. \tag{9}$$

The main idea, as proposed by Dijkstra himself, is to maintain ok in such a way that in each step of the algorithm the exact solution of the problem is known for target nodes in ok. This can be enforced by the invariant

$$dd(ok)\cdot ok = d\cdot ok \,. \tag{10}$$

For each node u outside ok the algorithm computes an approximation of a shortest path from s to u.

[2] In [Dij59], the set is called I.

The invariant holds trivially for $ok = \mathbf{0}$, i.e., we may initialise ok to $\mathbf{0}$. If, dually, $ok = \mathbf{1}$ we have solved the complete problem, because $dd(\mathbf{1}) = d$ by definition.

The goal of our derivation is to find an inductive version of dd that does not use star operations any more and extends the set ok in every step by at least one point until $ok = \mathbf{1}$ is achieved. Of course there is a priori no guarantee that $ok = \mathbf{1}$ can be achieved, since the set could have an infinite size. However, in all presented examples the set contains finitely many objects only, and so there we can guarantee termination.

The "strategy" is to extract maximal subexpressions of the form $p \cdot a \cdot p$ for a point p to allow application of the no-detours rule.

To support that further, we show that the invariant implies a kind of no-detours rule for ok:

Lemma 4.1. *If Equation* (10) *holds then* $dd(ok) \cdot e^* \cdot ok = dd(ok) \cdot ok$.

Proof. $dd(ok) \cdot ok \leq dd(ok) \cdot e^* \cdot ok$ holds by $\mathbf{1} \leq e^*$. The reverse inequation holds by

$$
\begin{array}{ll}
 & dd(ok) \cdot e^* \cdot ok \\
\leq & \quad \{\![\text{ (9) and isotony }]\!\} \\
 & d \cdot e^* \cdot ok \\
= & \quad \{\![\text{ definition of } d \ ]\!\} \\
 & s \cdot e^* \cdot e^* \cdot ok \\
= & \quad \{\![\ a^* \cdot a^* = a^* \text{ (KA) }]\!\} \\
 & s \cdot e^* \cdot ok \\
= & \quad \{\![\text{ definition of } d \ ]\!\} \\
 & d \cdot ok \\
= & \quad \{\![\text{ (10) }]\!\} \\
 & dd(ok) \cdot ok
\end{array}
$$

□

We now start with the derivation of an inductive definition of dd. For the **induction base** $ok = \mathbf{0}$ we calculate

$$dd(\mathbf{0}) \ = \ s \cdot \mathbf{0}^* \ = \ s \cdot \mathbf{1} \ = \ s \ .$$

The fact that $\mathbf{0}^* = \mathbf{1}$ follows immediately from the axioms of Kleene algebra.

At this point we see that one might also start with $ok = s$, since this, too, satisfies the Invariant (10). This follows, since for arbitrary $a \in S$ we have by distributivity, star unfold, neutrality of $\mathbf{1}$ and the no-detours rule,

$$s \cdot a^* \cdot s \ = \ s \cdot (\mathbf{1} + a \cdot a^*) \cdot s \ = \ s \cdot s + s \cdot a \cdot a^* \cdot s \ = \ s \ ,$$

so that

$$dd(s) \cdot s \ = \ s \cdot (s \cdot e)^* \cdot s \ = \ s \ = \ s \cdot e^* \cdot s \ = \ d \cdot s \ .$$

This reflects that the empty path with zero edges, represented by $\mathbf{1}$, is the shortest path from s to itself.

For the **induction step** we determine the behaviour of dd when ok is extended by a point $w \leq \neg ok$; from this we also infer how to choose w appropriately to maintain the invariant. First we calculate

$$
\begin{array}{ll}
 & dd(w + ok) \\
= & \quad \{\![\text{ definition of } dd \text{ and abbreviation } h =_{df} (w + ok) \cdot e \ ]\!\} \\
 & s \cdot h^* \\
= & \quad \{\![\text{ distributivity and grouping }]\!\} \\
 & s \cdot (ok \cdot e)^* \cdot (\mathbf{1} \ + \ w \cdot e \cdot h^*) \\
= & \quad \{\![\text{ definition of } dd \ ]\!\} \\
 & dd(ok) \cdot (\mathbf{1} \ + \ w \cdot e \cdot h^*) \ .
\end{array}
$$

The second summand can be expanded by the following calculation:

$$w \cdot e \cdot h^*$$

$$
\begin{array}{ll}
= & \quad \{\![\text{ star unfold and distributivity }]\!\} \\
& w \cdot e + w \cdot e \cdot h^* \cdot h \\
= & \quad \{\![\text{ definition of } h \text{ and distributivity }]\!\} \\
& w \cdot e + w \cdot e \cdot h^* \cdot w \cdot e + w \cdot e \cdot h^* \cdot ok \cdot e \\
= & \quad \{\![\text{ middle summand} \leq \text{first one by the no-detours rule }]\!\} \\
& w \cdot e + w \cdot e \cdot h^* \cdot ok \cdot e \ .
\end{array}
$$

By substituting the extended form back into the original expression we obtain the following refined formula:

$$dd(w + ok) \ = \ dd(ok) \cdot (\mathbf{1} \ + \ w \cdot e \ + \ w \cdot e \cdot h^* \cdot ok \cdot e) \ .$$

Next, we distribute the term $dd(ok)$ to the summands and continue our transformation with the third resulting summand:

$$
\begin{array}{ll}
& dd(ok) \cdot w \cdot e \cdot h^* \cdot ok \cdot e \\
\leq & \quad \{\![\text{ by } w \cdot e \leq e \text{ and } h \leq e \text{ (source restriction), and star unfold }]\!\} \\
& dd(ok) \cdot e^* \cdot ok \cdot e \\
= & \quad \{\![\text{ by the assumption that (10) holds and Lemma 4.1 }]\!\} \\
& dd(ok) \cdot ok \cdot e \\
\leq & \quad \{\![\text{ definition of } dd(ok) = s \cdot (ok \cdot e)^* \text{ and } a^* \cdot a \leq a^* \text{ (KA) }]\!\} \\
& dd(ok) \ ,
\end{array}
$$

which is the first summand. By definition of $\leq$ therefore it can be omitted. The informal interpretation of this is that shortest paths to nodes outside ok cannot loop back through ok, since this would add costs.

Altogether we have obtained the following version:

$$dd(w + ok) \ = \ dd(ok) \cdot (\mathbf{1} + w \cdot e) \ . \tag{11}$$

In Section 6, we will relate this equation to an assignment in a programming language, which works on nodes and uses path-weights to determine shortest paths.

We now want to choose w such that the invariant holds for $w + ok$ again. To this end we first calculate

$$
\begin{array}{ll}
& dd(w + ok) \cdot (w + ok) \\
= & \quad \{\![\text{ by (11) }]\!\} \\
& dd(ok) \cdot (\mathbf{1} + w \cdot e) \cdot (w + ok) \\
= & \quad \{\![\text{ by distributivity and neutrality of } \mathbf{1} \ ]\!\} \\
& dd(ok) \cdot (w + w \cdot e \cdot w) + dd(ok) \cdot (\mathbf{1} + w \cdot e) \cdot ok \\
= & \quad \{\![\text{ by the no-detours rule }]\!\} \\
& dd(ok) \cdot w + dd(ok) \cdot (\mathbf{1} + w \cdot e) \cdot ok \ .
\end{array}
$$

Second,

$$
\begin{array}{ll}
& dd(w + ok) \cdot (w + ok) \ = \ d \cdot (w + ok) \\
\Leftrightarrow & \quad \{\![\text{ by the above calculation and distributivity }]\!\} \\
& dd(ok) \cdot w + dd(ok) \cdot (\mathbf{1} + w \cdot e) \cdot ok \ = \ d \cdot w + d \cdot ok \\
\Leftrightarrow & \quad \{\![\text{ by } w \leq \neg ok, \ w \text{ and } ok \text{ are disjoint, hence (6) applies }]\!\} \\
& dd(ok) \cdot w \ = \ d \cdot w \ \wedge \ dd(ok) \cdot (\mathbf{1} + w \cdot e) \cdot ok \ = \ d \cdot ok \ .
\end{array}
$$

By distributivity and neutrality, the left hand side of the second conjunct equals

$$dd(ok) \cdot ok + dd(ok) \cdot w \cdot e \cdot ok \ ,$$

but by $w \cdot e \leq e \leq e^*$ and Lemma 4.1 we have

$$dd(ok) \cdot w \cdot e \cdot ok \leq dd(ok) \cdot e^* \cdot ok \ = \ dd(ok) \cdot ok \ ,$$

so that the second conjunct is equivalent to the invariant for ok and hence holds.

In sum, the invariant is maintained for $w + ok$ iff w is chosen to satisfy

$$dd(ok) \cdot w \;=\; d \cdot w \;.$$

This means that w must be chosen such that a shortest path from s to w can be obtained by extending some shortest path from s to an ok node by a single edge.

For abbreviation we now define $f =_{df} dd(ok) = s \cdot (ok \cdot e^*)$ and transform the right hand side further.

$$\begin{array}{ll} & d \cdot w \\ = & \quad \{\![\text{ definition of } d \;]\!\} \\ & s \cdot e^* \cdot w \\ = & \quad \{\![\text{ path grouping, using } e = ok \cdot e + \neg ok \cdot e \;]\!\} \\ & s \cdot (ok \cdot e^*) \cdot (\mathbf{1} + \neg ok \cdot e \cdot e^*) \cdot w \\ = & \quad \{\![\text{ definitions of } f \text{ and setting } e^+ =_{df} e \cdot e^* \;]\!\} \\ & f \cdot (\mathbf{1} + \neg ok \cdot e^+) \cdot w \\ = & \quad \{\![\text{ distributivity }]\!\} \\ & f \cdot w + f \cdot \neg ok \cdot e^+ \cdot w \;. \end{array}$$

By Invariant (10) for ok therefore the invariant for $ok + w$ is equivalent to $f \cdot w \;=\; f \cdot w + f \cdot \neg ok \cdot e^+ \cdot w$ and hence to

$$f \cdot \neg ok \cdot e^+ \cdot w \;\;\leq\;\; f \cdot w \;.$$

By the no-detours-rule this is implied by

$$f \cdot \neg ok \;\leq\; f \cdot w \cdot \top \;,$$

where we assume that the underlying i-semiring has a greatest element $\top$, which in the case of the path sets is $\top = \Sigma + \Sigma^2$ (see Section 3). We need the factor $\top$ to make the formula usable in the matrix model mentioned in Proposition 2.1. The reason is the following. The matrix corresponding to $f \cdot w$ alone has non-trivial entries only in the column for w, whereas the one corresponding to $f \cdot \neg ok$ has non-trivial entries only in the $\neg ok$ columns. Therefore the comparison $f \cdot \neg ok \leq f \cdot w$ is almost always false. By multiplying with $\top$, the row for w is copied to all columns and can then sensibly be used for comparison with the costs for the nodes in $\neg ok$.

For better understanding we now bring this into pointwise form, assuming that in the underlying cost semiring S the test set is *point-generated*, i.e., every test different from $\mathbf{0}$ is the sum of all points below it, which holds, e.g., in UP(Σ), but also in every test-discrete semiring:

$$\begin{array}{ll} & f \cdot \neg ok \leq f \cdot w \cdot \top \\ \Leftrightarrow & \quad \{\![\text{ splitting } \neg ok \text{ into its points }]\!\} \\ & \sum\limits_{v \leq \neg ok} f \cdot v \leq f \cdot w \cdot \top \\ \Leftrightarrow & \quad \{\![\text{ universal characterisation (5) of choice }]\!\} \\ & \forall\, v \leq \neg ok : f \cdot v \leq f \cdot w \cdot \top \;. \end{array}$$

This property, below abbreviated by $ddmin(w, \neg ok)$, holds iff w is a point of $\neg ok$ with minimal $dd(ok)$-value.

From all our results we can now assemble the complete algorithm:

$$\begin{array}{rcl} dd(\mathbf{0}) & = & s \\ dd(ok + w) & = & dd(ok) \cdot (\mathbf{1} + w \cdot e) \end{array}$$
provided $ok \neq \mathbf{0}$ and point $w \leq \neg ok$ satisfies $ddmin(w, \neg ok)$.

This is the classical version of the algorithm as known from the textbooks: in each step, choose a point outside ok with minimal distance to the start node s; for all other nodes v outside ok adapt the distance when use of the newly available node w provides a shortcut relatively to the current shortest paths from s to v. This potential adaptation is algebraically reflected by the term $\mathbf{1} + w \cdot e$.

5. The Floyd/Warshall Algorithm

We show the algebra at work in a derivation of an algorithm for the all-pairs shortest non-empty path problem, which has the computation of the transitive closure of a matrix or relation as a special case. The specification is even simpler than that for Dijkstra's algorithm: given an element e of an arbitrary Kleene algebra,

Compute e^+ . (12)

The aim of the derivation is to obtain the algorithm of Floyd/Warshall. Of course, the reflexive-transitive closure and the transitive closure are interdefinable as $e^* = \mathbf{1} + e^+$ and $e^+ = e \cdot e^*$.

The central idea is again to use a set ok that restricts the inner nodes of the paths under consideration and to increment it stepwise. Therefore we specify an auxiliary function (rt for "restricted transitive closure"):

$rt(ok) =_{df} e \cdot (ok \cdot e)^*$.

For the **induction base** $ok = \mathbf{0}$ we obtain

$rt(\mathbf{0}) = e \cdot \mathbf{0}^* = e \cdot \mathbf{1} = e$.

To prepare the **induction step** we calculate, for *arbitrary* point w:

$$
\begin{array}{ll}
 & rt(ok + w) \\
= & \quad \{\![\text{ definition } rt \text{ and distributivity }]\!\} \\
 & e \cdot (ok \cdot e + w \cdot e)^* \\
= & \quad \{\![\text{ star of sum }]\!\} \\
 & e \cdot (ok \cdot e)^* \cdot (w \cdot e \cdot (ok \cdot e)^*)^* \\
= & \quad \{\![\text{ fold } e \cdot (ok \cdot e)^* \text{ twice to } f =_{df} rt(ok)]\!\} \\
 & f \cdot (w \cdot f)^* \\
= & \quad \{\![\text{ star recursion and distributivity }]\!\} \\
 & f \ + \ f \cdot w \cdot f \cdot (w \cdot f)^* \\
= & \quad \{\![\text{ star recursion and distributivity }]\!\} \\
 & f \ + \ f \cdot w \cdot f \ + \ f \cdot w \cdot f \cdot (w \cdot f)^* \cdot w \cdot f \\
= & \quad \{\![\text{ since third alternative} \leq \text{ second one by no-detours rule }]\!\} \\
 & f \ + \ f \cdot w \cdot f \ .
\end{array}
$$

To guarantee termination, one should choose $w \leq \neg ok$ so that ok increases by one point in every step.

The complete algorithm now reads as follows:

$$
\begin{array}{rcl}
rt(\mathbf{0}) & = & e \\
rt(ok + w) & = & f \ + \ f \cdot w \cdot f \\
 & & \text{where } f = rt(ok) \text{ and } w \notin ok \ .
\end{array}
$$

Depending on the underlying cost semiring (see Section 6.1) this is the Floyd or Warshall algorithm.

6. Applications

Since we have lifted the derivation of the algorithms Dijkstra and Floyd/Warshall to an abstract algebraic level, we are now able to replace the underlying default cost semiring, in which the cost of a path is the number of its edges, by an arbitrary one.

6.1. More Refined Cost Models

In Section 2 we have just used the number of adjacent edges in a path as the measure of its length. Frequently one needs a different measure, where costs of single edges are e.g. non-negative reals (representing edge weights), or Booleans (representing connectivity only). From the literature (e.g. [GM08]) it is well known

that algebraically this can again be cast into semiring terminology. The semirings used are special cases of cost semirings as defined in Section 3. In this section we look at some of the most common semirings from the literature and briefly discuss their applications when used in combination with the algorithms presented above.

Some classical examples are the following.

Example 6.1.

1. The set $\mathbb{B} = \{\mathtt{false}, \mathtt{true}\}$ of Booleans forms, together with the Boolean connectives, an i-semiring $(\mathbb{B}, \vee, \wedge, \mathtt{false}, \mathtt{true})$ in which the natural order is the implication order. Since it has only the two elements $\mathbf{0}$ and $\mathbf{1}$ it is trivially test-discrete and hence a cost semiring. It can be used to "measure" reachability or connectivity between nodes.
2. The *tropical i-semiring* [GM08] is given by $(min, +) =_{df} (\mathbb{N}_\infty, min, +, \infty, 0)$, where $\mathbb{N}_\infty =_{df} \mathbb{N} \cup \{\infty\}$. Note that *min* and ∞ play the rôles of semiring $+$ and $\mathbf{0}$, while addition $+$ of extended natural numbers and 0 play the rôles of semiring $\cdot$ and $\mathbf{1}$. The natural ordering in this i-semiring is the converse of the standard ordering on $\mathbb{N}_\infty$, in particular, $\mathbf{1} = 0$ is the largest element. Therefore all elements are potential tests. However, since the order is linear, only the semiring $\mathbf{0}$ and $\mathbf{1}$ have relative complements, i.e., the tropical i-semiring is test-discrete, and it follows by (7) that it is a cost semiring. It captures the cost measure "edge number" of a path, which we have been using in the earlier sections. A standard implementation technique here is to "complete" the given graph to a clique by adding an edge with cost ∞ between every two nodes which are not yet connected by an edge. This avoids a case distinction on the existence of edges.
   For example, over such a completed graph, Equation (11)
   $$dd(w + ok) \;=\; dd(ok) \cdot (\mathbf{1} + w \cdot e)$$
   can now be interpreted as follows. The right hand side is the algebraic equivalent of the usual set of assignments
   $$dd[v] = min(dd[v], dd[w] + weight(w, v)) \;, \tag{13}$$
   for all nodes $v \notin ok$, in a programming language, where the function *weight* returns the cost of an edge.
3. Similarly, the structure $(\mathbb{R}^{\geq 0}_\infty, min, +, \infty, 0)$ is a cost semiring, where $\mathbb{R}^{\geq 0}_\infty =_{df} \{r \in \mathbb{R} \mid r \geq 0\} \cup \{\infty\}$.
4. The *max-min-i-semiring* [Car79] or *bottleneck algebra* is $(\mathbb{R}^{\geq 0}_\infty, max, min, 0, \infty)$. The elements denote capacities, e.g., of pipes. It is used in determining the maximal throughput in a network. While *min* determines the "bottleneck" along a path, *max* chooses the paths with more throughput. This is also a cost semiring, since it is again linearly ordered and hence test-discrete.

□

Elements of cost semirings can be used as edge labels. An *edge-labelled graph* over a set Σ of nodes is a set $E \subseteq \Sigma^2$ of edges together with a *weight function* $W : E \to S$, where S is some cost semiring and $W(u, u) = \mathbf{1}$ when $(u, u) \in E$. As in Example 6.1.2, one can extend W to a total function from Σ^2 to S by setting $W(u, v) =_{df} \mathbf{0}$ when $(u, v) \notin E$. We will assume this extended version in the sequel.

A path in that graph is defined as before, with the restriction that adjacent nodes must form edges in E. The *cost* of a path p is now the S-sum of the weights of its edges. More formally,

$$cost(p) =_{df} \begin{cases} \mathbf{1} & \text{if } p \in \Sigma \text{ is a node} \\ \sum_{i=0}^{n-1} W(v_i.v_{i+1}) & \text{if } p = v_0 \dots v_n \;. \end{cases}$$

The first case is necessary to satisfy the no detours rule. All other definitions given in Section 2 remain the same, so that we obtain right away an algebra of uniform sets of paths with more general edge weights than unit cost. In particular, the natural order in these semirings is used to distinguish minimum-cost edges. Still, we are using basically the path set model $\mathrm{UP}(\Sigma)$ over a set Σ of nodes. There the costs, of course, influence the operations on paths, but are not made explicit.

By Theorem 3.4 and the discussion following it, each cost semiring presented can be lifted to another semiring at the matrix level. The set $\mathrm{MAT}(M, \mathbb{B})$ of Boolean matrices, for example, is again a cost semiring; it is isomorphic to the i-semiring $\mathrm{REL}(M)$ of binary relations over M under union and composition.

As shown in the previous sections the algorithms of Dijkstra and Floyd/Warshall require only a cost-semiring. Hence we can use any of the presented cost-semirings for the algorithms. The outcomes, of course,

no longer determine the shortest paths in terms of the edge numbers but cost-minimal ones in terms of the respective semiring order.

Example 6.2.

1. When applying $(\mathbb{B}, \vee, \wedge, \texttt{false}, \texttt{true})$, or more precisely $\mathrm{MAT}(M, \mathbb{B})$, to Dijkstra's algorithm, it yields immediately an algorithm for connectivity and for determining the reflexive-transitive closure; of course Floyd/Warshall's algorithms yield the transitive closure only.
2. The *tropical i-semiring* or $(\mathbb{R}_{\geq 0} \cup \{\infty\}, \mathit{min}, +, \infty, 0)$ as underlying structure, can be used to determine the lengths of shortest paths without storing the paths themselves.
3. The *bottleneck algebra* does not store paths either, it only keeps track of the capacity of a path, which is immediately modelled by the *min-max*-semiring. This yields immediately an algorithm for solving flow problems.

□

Many interesting further cost models can be found in the papers [LT91b, LT91a].

6.2. Routing protocols

Often, in particular in network routing, the protocol/algorithm is not intended to take track of the entire path with its costs; it is sufficient that every node knows its *precursor*, which is the last node but one on the considered path to that particular node. When we store one precursor with each node (except the start node), information about the entire path can be extracted—just go from the target to its precursor, than to the precursor of the precursor and so on.

Formally, the *precursor* of the end node on a path $p = v_0 \ldots v_{n-1}.v_n$ with $n > 0$ is v_{n-1}.

We define a cost semiring that administers conventional path costs together with precursor nodes. For the former we assume a linearly ordered cost semiring $S = (S, +, \cdot, \mathbf{0}, \mathbf{1})$. As before, $\mathbf{0}$ represents the cost of an infinite (impossible) path. Additionally we assume that costs are cancellative, i.e., $c \cdot d = c \cdot e \Rightarrow d = e$ for all $c \neq 0$. This requirement is not very restrictive and is for example satisfied by $(\mathbb{N}, +, 0)$. The set Σ of nodes is used to describe precursors. We only look at uniform paths in combination with precursors, hence we can calculate in the set $\mathcal{P}(\Sigma) \times S$. An element (PP, c) lists precursors in its first component and the cost of the paths via the precursors in its second component. Due to this interpretation the pairs $(\mathrm{PP}, \mathbf{0})$ with $\mathrm{PP} \neq \emptyset$ do not make sense: if there is an unsurmountable obstacle of infinite cost, there cannot be a path to the destination and therefore the set of precursor must be empty. Similarly, we will exclude paths $(\mathrm{PP}, \mathbf{1})$ with $\mathrm{PP} \neq \emptyset$: no paths via precursors can have trivial cost. That means, we only use

$$\begin{array}{l}\text{pairs } (\mathrm{PP}, \mathbf{0}) \text{ with } \mathrm{PP} = \emptyset \text{ , and} \\ \text{pairs } (\mathrm{PP}, \mathbf{1}) \text{ with } \mathrm{PP} = \emptyset \text{ .}\end{array} \tag{14}$$

For the set $\mathrm{U\Sigma C} =_{df} \{(\emptyset, \mathbf{0}), (\emptyset, \mathbf{1})\} \cup (\mathcal{P}(\Sigma) \times (S - \{\mathbf{0}, \mathbf{1}\}))$, we define a multiplication $\otimes$ by

$$(\mathrm{PP}, c) \otimes (\mathrm{QQ}, d) =_{df} \begin{cases} (\mathrm{PP}, c \cdot d) & \text{if } \mathrm{PP} \neq \emptyset \wedge (\mathrm{QQ}, d) \neq (\emptyset, \mathbf{0}) \\ (\mathrm{QQ}, c \cdot d) & \text{if } \mathrm{PP} = \emptyset \wedge c \cdot d \neq \mathbf{0} \\ (\emptyset, \mathbf{0}) & \text{otherwise .} \end{cases}$$

It follows that $(\mathrm{U\Sigma C}, \otimes, (\emptyset, \mathbf{1}))$ is a monoid and $(\emptyset, \mathbf{0})$ an annihilator (see appendix).

Choice on elements $(\mathrm{PP}, c), (\mathrm{QQ}, d) \in \mathrm{U\Sigma C}$ is defined as follows.

$$(\mathrm{PP}, c) \oplus (\mathrm{QQ}, d) =_{df} \begin{cases} (\mathrm{PP}, c) & \text{if } c > d \\ (\mathrm{QQ}, d) & \text{if } d > c \\ (\mathrm{PP} \cup \mathrm{QQ}, c) & \text{if } c = d \text{ .} \end{cases}$$

Note that here $>$ refers to the general semiring order on S, in which the greater elements are the better ones.

Proposition 6.3. *The structure* $(\mathrm{U\Sigma C}, \oplus, \otimes, (\emptyset, \mathbf{0}), (\emptyset, \mathbf{1}))$ *is a cost semiring (with point-generated test set) when* S *is, and therefore so is the matrix semiring over* $\mathrm{U\Sigma C}$.

We represent an edge-labelled graph with edge set $E \subseteq \Sigma^2$ and weight function $W : E \to S$ by a matrix

M with entries in UΣC as follows: for $u, v \in \Sigma$,

$$M_{uv} =_{df} \begin{cases} (\emptyset, \mathbf{1}) & \text{if } u = v \\ (\{u\}, W(u.v)) & \text{if } u \neq v \ . \end{cases}$$

Now we can use our general Dijkstra algorithm in the matrix semiring over UΣC to compute precursors and costs simultaneously while avoiding explicit storage of the associated paths.

Let again s be the starting node. Then we record for each node $n \neq s$ and its corresponding entry $(\mathrm{PP}_{sn}, c_{sn}) =_{df} dd(ok)_{sn}$ in the matrix $dd(ok)$ the components in two arrays pre and dd, i.e., $(pre[n], dd[n]) = (\mathrm{PP}_{sn}, c_{sn})$. With this, the implementation reads in pseudocode as follows.

```
dd[s] = 1;
pre[s] = ∅;
for all (v ≠ s)
{ dd[v] = W(s.v);
  pre[v] = {s};
}
ok = {s};
while (ok ≠ Σ)
{ find w with w ∉ ok and maximal value dd[w];
  ok = ok ∪ {w};
  for all (v ∉ ok)
  { cost c = dd[w] · W(w.v);
    if (c > dd[v])
       { dd[v] = c;
         pre[v] = {w};
       }
    else if (c == dd[v])
            pre[v] = pre[v] ∪ {w};
  }
}
```

7. Related Work

The present paper is, of course, not the first one by far to deal with an algebraic approach to correctness proofs or refinement-style derivations of Dijkstra's algorithm and its relatives. One of the earliest papers on that is by Carré [Car71], in which he introduces the semiring operations as well as the matrix semiring, but not yet the Kleene star. He attributes the notation for the semiring operations to [Ber62]. Differing from later papers, he requires semiring multiplication to be commutative, which excludes operations such as path fusion, but allows, e.g., to sum up costs. He also introduces a partial order which is the converse of the natural semiring order. He then develops methods for solving equations over the matrix semiring and applies them to quite a number of problems, among them the Floyd/Warshall algorithm and Dantzig's method [Dan60] for solving shortest-route problems, but not yet to Dijkstra's algorithm.

His approach has been taken up and refined in [BC75]. What is called a regular algebra there corresponds closely to Kleene algebra as used in our paper; in particular, semiring multiplication is no longer required to be commutative. However, the axioms do characterise the Kleene star uniquely only for so-called definite elements i.e., elements a that satisfy $x = a \cdot x \Rightarrow x = 0$. Still, Dijkstra's algorithm is not treated.

The first semiring-based correctness proof of Dijkstra's algorithm we are aware of is given in [AHU74]. There the Kleene star is not characterised by axioms of its own but rather defined as an infinite sum in a closed semiring (similar to the definition of reflexive transitive closure in Section 2), a weaker version of the so-called standard Kleene algebras introduced in [Con71].

Two very comprehensive books with algebraic treatments of graphs algorithms are [Car79] and [GM79], the latter republished as [GM88] and recently in substantially extended and revised form as [GM08]. In addition to their abstract-algebraic material, both sets of authors present a very simple technique for solving

equations of the form $x = A \cdot x + b$, where A and b are a matrix and a vector over a cost semiring in which $\mathbf{1}$ is the greatest element. The shape of this equation is also known from Bellman's approach to routing problems [Bel58]. The steps for eliminating iteration perform the same transformations as the assignments (13) above. Since in both settings the least solution of the above equation is $x = A^* \cdot b$ (although star is defined a bit differently than in Kleene algebra, meaning iteration till a fixpoint is reached), these elimination procedures can be used to solve the shortest-problem as well. Hence both books provide early calculational approaches to Dijkstra's algorithm, albeit in a relatively concrete algebraic setting.

A unified treatment of several path problems in graphs, based on a more concrete semiring of regular expressions, is given in [Tar81].

Then, for some time, the topic seems to have lost interest. It was resumed in another fully calculational approach to the derivation of the algorithms by Dijkstra and Floyd/Warshall in [BEG94], working again in the particular regular algebra of matrices, but in a much more algebraic style than [Car79].

We see the following main innovations of our approach relative to that paper and the earlier ones. First, we are working completely in abstract semiring theory without resorting to a specific model such as the matrix one. In particular, sets of nodes are abstractly represented as semiring tests, and single nodes as points, i.e., atomic tests. There is no need for special selectors to extract elements out of matrices; they are uniformly represented by simple pre- and postmultiplication with points. This simplifies many calculations. Second, the characteristic property which is at the heart of all cost algebras, namely the no-detours rule, can be given in a more general fashion using again the notion of a point. It also works for matrix semirings, and the more usual requirement that the multiplicative semiring unit is the greatest element (which does not work in the matrix semiring) falls out as a special case for cost semirings that are linearly ordered. We have taken advantage of the additional generality by defining and deploying several unconventional cost semirings.

8. Conclusion

This paper sheds an algebraic light on Dijkstra's shortest path algorithm and on the algorithm of Floyd/Warshall. Based on preliminary work of Carré, Backhouse and others, we have combined these problems with Kleene algebra and have presented a purely algebraic derivation for the algorithm. This was done by starting from abstract specifications of the problems followed by step-by-step refinements. Extending earlier algebraic approaches, the derivation is completely independent of a particular underlying model, which leads to much greater generality and flexibility. We have also pinpointed the central rôle played by special elements called *points* and the associated *no-detours rule*, which abstractly states that the best path from a point to itself always is the one with zero edges. As a sample application of the theory, we have defined a new algebra of path measures particularly suited to an application to routing protocols.

To further assess the merits of the approach, we discuss some relationships between Dijkstra's shortest path algorithm and routing protocols in networks in more detail. Due to these close relationships, the algebraic constructions presented in this paper will help to understand and to analyse the structure of these protocols. A detailed examination is part of our future work; here we just point out the potential of algebra.

Both the *Intermediate System To Intermediate System* (IS-IS) [Ora90] and the *Open Shortest Path First* (OSPF) [Moy98, CFM08] routing protocols use the same Dijkstra algorithm for computing the best path through the network (based on the path length). Due to this, the correctness of the protocols with respect to the shortest paths found can be immediately transferred from the correctness of Dijkstra's algorithm or even from the derivation given in this paper. Both protocols work on wireless networks such as wireless mesh networks and mobile ad-hoc network (MANET).

More intrinsic protocols for wireless networks, such as the *Ad Hoc On-Demand Distance Vector* (AODV) [PBD03] routing protocol, use precursors and path lengths to determine the best route. Hence one can use our matrix-based model over pairs as a basic approximation for these protocols. A first analysis w.r.t. AODV has been done in [HM11]. However, since reasonably complex routing protocols use, next to precursors and path lengths, other attributes like sequence numbers (indicating the freshness of a route), the presented approach has to be extended to cover these properties.

Acknowledgment We are grateful to Roland Glück for a number of valuable remarks on a preliminary version of this paper. We also thank the anonymous referees for the careful reading and their suggestions, which led to considerable streamlining of the paper.

References

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[BC75] R. Backhouse and B. Carré. Regular algebra applied to path-finding problems. *Journal of the Institute of Mathematics and Applications*, 1975.

[BEG94] R. Backhouse, J. van den Eijnde, and A. van Gasteren. Calculating path algorithms. *Science of Computer Programming*, 22(1–2):3–19, 1994.

[Bel58] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[Ber62] C. Berge. *The Theory of Graphs and its Applications*. Methuen, 1962.

[Car71] B. Carré. An algebra for network routing problems. *IMA Journal of Applied Mathematics*, 7:273–294, 1971.

[Car79] B. Carré. *Graphs and Networks*. Oxford Applied Mathematics & Computing Science Series. Clarendon Press, Oxford University, 1979.

[CFM08] R. Coltun, D. Ferguson, and J. Moy. OSPF for IPv6. RFC 2328 (Standard, Errata Exist), 2008.

[Con71] J. Conway. *Regular Algebra and Finite Machines*. Chapman & Hall, 1971.

[Dan60] G. Dantzig. On the shortest route through a network. *Management Science*, 6:187–190, 1960.

[Dij59] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[DP02] B. Davey and H. Priestley. *Introduction to lattices and order,*. Cambridge University Press, 2nd edition, 2002.

[Flo62] R. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5:345, 1962.

[GM79] M. Gondran and M. Minoux. *Graphes et Algorithmes*. Eyrolles, 1979.

[GM88] M. Gondran and M. Minoux. *Graphs and Algorithms*. Wiley, 1988.

[GM08] M. Gondran and M. Minoux. *Graphs, Dioids and Semirings — New Models and Algorithms*. Springer, 2008.

[HM11] P. Höfner and A. McIver. Towards an algebra of routing tables. In H. de Swart, editor, *Relations and Kleene Algebra in Computer Science*, volume 6663 of *Lecture Notes in Computer Science*, pages 212–229. Springer, 2011.

[Hol98] M. Hollenberg. Equational axioms of test algebra. In M. Nielsen and W. Thomas, editors, *CSL '97: Selected Papers from the 11th International Workshop on Computer Science Logic*, volume 1414 of *Lecture Notes in Computer Science*, 1998.

[HS07] P. Höfner and G. Struth. Automated reasoning in Kleene algebra. In F. Pfennig, editor, *Automated Deduction*, volume 4603 of *Lecture Notes in Artificial Intelligence*, pages 279–294. Springer, 2007.

[Kle51] S. Kleene. Representation of events in nerve nets and finite automata. Technical Report RM-704, RAND Corporation, 1951. RAND Research Memorandum.

[Kle52] S. C. Kleene. *Introduction to metamathematics*. Van Nostrand, 1952.

[Koz94] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.

[Koz97] D. Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems*, 19(3):427–443, 1997.

[LT91a] T. Lengauer and D. Theune. Efficient algorithms for path problems with general cost citeria. In J. Leach Albert, B. Burkhard Monien, and M. Rodríguez-Artalejo, editors, *Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8–12, 1991, Proceedings*, volume 510 of *Lecture Notes in Computer Science*, pages 314–326. Springer, 1991.

[LT91b] T. Lengauer and D. Theune. Unstructured path problems and the making of semirings (preliminary version). In F. Dehne, J.-R. Sack, and N. Santoro, editors, *Algorithms and Data Structures, 2nd Workshop WADS '91, Ottawa, Canada, August 14-16, 1991, Proceedings*, volume 519 of *Lecture Notes in Computer Science*, pages 189–200. Springer, 1991.

[Moy98] J. Moy. OSPF version 2. RFC 2328 (Standard, Errata Exist), 1998.

[Ora90] D. Oran. IS-IS protocol specification (IETF). RFC 1142 (Informational), February 1990.

[PBD03] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. RFC 3561 (Experimental), July 2003.

[Roy59] B. Roy. Transitivité et connexité. *Comptes Rendus de l'Académie des Sciences Paris*, 249:216–218, 1959.

[Tar81] R. Tarjan. A unified approach to path problems. *Journal of the ACM*, 28:577–593, 1981.

[War62] S. Warshall. A theorem on boolean matrices. *Journal of the ACM*, 9(1):11–12, 1962.

A. Omitted Proofs

In this appendix, we list all omitted proofs, except the ones for Propositions 3.1 and 3.2. These proofs are straightforward calculations and similar to the one for Proposition 6.3, which we present below.

We start by showing the first of the two (6)-splitting rules; the other can be proved similarly. $a \cdot p = c \cdot p \wedge b \cdot q = d \cdot q \Rightarrow a \cdot p + b \cdot q = c \cdot p + d \cdot q$ follows immediately by the Leibniz rule. The converse implication can be shown by idempotence of tests and the assumption:

$$
\begin{array}{ll}
 & a \cdot p + b \cdot q = c \cdot p + d \cdot q \\
\Leftarrow & \quad \{\![\text{ isotony }]\!\} \\
 & a \cdot p \cdot p + b \cdot q \cdot p = c \cdot p \cdot p + d \cdot q \cdot p
\end{array}
$$

$$
\begin{array}{ll}
\Leftarrow & \{\![\text{ idempotemce and assumption } (q \cdot p = \mathbf{0})]\!\} \\
& a \cdot p + b \cdot \mathbf{0} = c \cdot p + d \cdot \mathbf{0} \\
\Leftrightarrow & \{\![\text{ annihilation } a \cdot \mathbf{0} = \mathbf{0}]\!\} \\
& a \cdot p = c \cdot p \ .
\end{array}
$$

The implication $a \cdot p + b \cdot q = c \cdot p + d \cdot q \ \Rightarrow \ b \cdot q = d \cdot q$ can be shown by similar arguments, multiplying with the test q. □

Next we prove the star of sum rule (a proof can also be found in [Koz94]). We split the equation into two inequations. The equation $a^* \cdot (b \cdot a^*)^* \leq (a + b)^*$ follows from isotony. For the reverse inequation we use star induction:

$$
\begin{array}{ll}
& (a + b)^* \leq a^* \cdot (b \cdot a^*)^* \\
\Leftarrow & \{\![\text{ star induction }]\!\} \\
& (a + b) \cdot a^* \cdot (b \cdot a^*)^* + \mathbf{1} \leq a^* \cdot (b \cdot a^*)^* \\
\Leftrightarrow & \{\![\text{ distributivity and (5) }]\!\} \\
& a \cdot a^* \cdot (b \cdot a^*)^* \leq a^* \cdot (b \cdot a^*)^* \ \wedge \ b \cdot a^* \cdot (b \cdot a^*)^* + \mathbf{1} \leq a^* \cdot (b \cdot a^*)^* \\
\Leftarrow & \{\![\ a \cdot a^* \leq a^* \text{ (KA) on the left and (star unfold) on the right }]\!\} \\
& a^* \cdot (b \cdot a^*)^* \leq a^* \cdot (b \cdot a^*)^* \ \wedge \ (b \cdot a^*)^* \leq a^* \cdot (b \cdot a^*)^* \\
\Leftrightarrow & \{\![\ \mathbf{1} \leq a^* \text{ (KA) }]\!\} \\
& \texttt{true}
\end{array}
$$

Using this we prove the grouping rule:

$$(b + c)^* \ = \ c^* \cdot (b \cdot c^*)^* \ = \ c^* \cdot (\mathbf{1} + b \cdot c^* \cdot (b \cdot c^*)^*) \ = \ c^* \cdot (\mathbf{1} + b \cdot (b + c)^*) \ .$$

The second steps uses star unfold; the second equation of the grouping rule is proved analogously. □

Before proving Proposition 6.3, we discuss two consequences of cancellativity. First, by definition of $\leq$, cancellativity also holds for inequalities. That means for an arbitrary i-semiring $(S, +, \cdot, \mathbf{0}, \mathbf{1})$ with $c \cdot d = c \cdot e \ \Rightarrow \ d = e$ ($c, d, e \in S$ and $c \neq \mathbf{0}$), we have, for all $c, d, e \in S$,

$$c \cdot d > c \cdot e \ \ \Rightarrow \ \ d > e \ .$$

For $c \neq \mathbf{0}$ we obtain even an equivalence:

$$c \neq \mathbf{0} \ \ \Rightarrow \ \ (c \cdot d > c \cdot e \ \ \Leftrightarrow \ \ d > e) \ . \tag{15}$$

Cancellativity of $\cdot$ also guarantees the absence of zero divisors, i.e., for all $a, b \in S$

$$a \cdot b = \mathbf{0} \ \ \Rightarrow \ \ a = \mathbf{0} \vee b = \mathbf{0} \ . \tag{16}$$

Therefore, $a \cdot b = \mathbf{0} \wedge a \neq \mathbf{0} \Rightarrow b \Rightarrow \mathbf{0}$.

Now we can prove Proposition 6.3. By the definition of $\otimes$, it is easy to see that $(\emptyset, \mathbf{0})$ is an annihilator

$$(\mathrm{PP}, c) \otimes (\emptyset, \mathbf{0}) \ = \ (\emptyset, \mathbf{0}) \otimes (\mathrm{QQ}, d) \ = \ (\emptyset, \mathbf{0}) \ .$$

Next we show that $(\mathrm{U\Sigma C}, \otimes, (\emptyset, \mathbf{1}))$ is a monoid. $(\emptyset, \mathbf{1})$ is the neutral element since

$$(\emptyset, \mathbf{1}) \otimes (\mathrm{QQ}, d) = \left\{ \begin{array}{ll} (\mathrm{QQ}, \mathbf{1} \cdot d) & \text{if } \mathbf{1} \cdot d \neq \mathbf{0} \\ (\emptyset, \mathbf{0}) & \text{if } \mathbf{1} \cdot d = \mathbf{0} \end{array} \right\} = \left\{ \begin{array}{ll} (\mathrm{QQ}, d) & \text{if } d \neq \mathbf{0} \\ (\emptyset, \mathbf{0}) & \text{if } d = \mathbf{0} \end{array} \right\} \overset{(14)}{=} (\mathrm{QQ}, d)$$

$$(\mathrm{PP}, c) \otimes (\emptyset, \mathbf{1}) = \left\{ \begin{array}{ll} (\mathrm{PP}, c) & \text{if } \mathrm{PP} \neq \emptyset \\ (\emptyset, c) & \text{if } \mathrm{PP} = \emptyset \wedge c \neq \mathbf{0} \\ (\emptyset, \mathbf{0}) & \text{if } \mathrm{PP} = \emptyset \wedge c = \mathbf{0} \end{array} \right\} = (\mathrm{PP}, c)$$

To simplify the proof of associativity, we first note that for any algebra with an operator $\odot$ we automatically have $(a \odot b) \odot c = a \odot (b \odot c)$ whenever one of a, b, c is a neutral element or an annihilator w.r.t. $\odot$. Therefore in our concrete precursor algebra we only need to analyse the elements different from $(\emptyset, \mathbf{0})$ and $(\emptyset, \mathbf{1})$. Since we assume cancellativity and hence absence of zero divisors, for such elements the definition of

$\otimes$ simplifies to

$$(\mathrm{PP}, c) \otimes (\mathrm{QQ}, d) \;=\; \begin{cases} (\mathrm{PP}, c \cdot d) & \text{if } \mathrm{PP} \neq \emptyset \\ (\mathrm{QQ}, c \cdot d) & \text{otherwise .} \end{cases} \tag{17}$$

So assume $(\mathrm{PP}, c), (\mathrm{QQ}, d), (\mathrm{RR}, e) \notin \{(\emptyset, \mathbf{0}), (\emptyset, \mathbf{1})\}$.

$$\begin{array}{ll} & ((\mathrm{PP}, c) \otimes (\mathrm{QQ}, d)) \otimes (\mathrm{RR}, e) \\ = & \quad \{\![\ (17)\ ]\!\} \\ & \begin{cases} (\mathrm{PP}, c \cdot d) \otimes (\mathrm{RR}, e) & \text{if } \mathrm{PP} \neq \emptyset \\ (\mathrm{QQ}, c \cdot d) \otimes (\mathrm{RR}, e) & \text{if } \mathrm{PP} = \emptyset \end{cases} \\ = & \quad \{\![\ (17),\ \text{associativity of } \cdot\ ]\!\} \\ & \begin{cases} (\mathrm{PP}, c \cdot d \cdot e) & \text{if } \mathrm{PP} \neq \emptyset \\ (\mathrm{QQ}, c \cdot d \cdot e) & \text{if } \mathrm{PP} = \emptyset \wedge \mathrm{QQ} \neq \emptyset \\ (\mathrm{RR}, c \cdot d \cdot e) & \text{if } \mathrm{PP} = \emptyset \wedge \mathrm{QQ} = \emptyset \end{cases} \\ = & \quad \{\![\ \text{splitting the first case}\ ]\!\} \\ & \begin{cases} (\mathrm{PP}, c \cdot d \cdot e) & \text{if } \mathrm{PP} \neq \emptyset \wedge \mathrm{QQ} \neq \emptyset \\ (\mathrm{PP}, c \cdot d \cdot e) & \text{if } \mathrm{PP} \neq \emptyset \wedge \mathrm{QQ} = \emptyset \\ (\mathrm{QQ}, c \cdot d \cdot e) & \text{if } \mathrm{PP} = \emptyset \wedge \mathrm{QQ} \neq \emptyset \\ (\mathrm{RR}, c \cdot d \cdot e) & \text{if } \mathrm{PP} = \emptyset \wedge \mathrm{QQ} = \emptyset \end{cases} \\ = & \quad \{\![\ \text{combining cases 1 and 3 as well as 2 and 4, (17)}\ ]\!\} \\ & \begin{cases} (\mathrm{PP}, c) \otimes (\mathrm{QQ}, d \cdot e) & \text{if } \mathrm{QQ} \neq \emptyset \\ (\mathrm{PP}, c) \otimes (\mathrm{RR}, d \cdot e) & \text{if } \mathrm{QQ} = \emptyset \end{cases} \\ = & \quad \{\![\ (17)\ ]\!\} \\ & (\mathrm{PP}, c) \otimes ((\mathrm{QQ}, d) \otimes (\mathrm{RR}, e)) \ . \end{array}$$

The proof that $(\mathrm{U}\Sigma\mathrm{C}, \oplus, (\emptyset, \mathbf{0}))$ forms a commutative and idempotent monoid is similar, but much simpler. We leave it to the interested reader.

For distributivity, we can apply a similar simplification technique as above. Consider a structure with operators $+$ and $\cdot$ with neutral elements $\mathbf{0}$ and $\mathbf{1}$ such that $\mathbf{0}$ also is an annihilator for $\cdot$. Then the equation $a \cdot (b + c) = a \cdot b + a \cdot c$ automatically holds when $a \in \{\mathbf{0}, \mathbf{1}\}$ or when $b = \mathbf{0}$ or $c = \mathbf{0}$. Hence in the precursor algebra we only need to analyse the case where $(\mathrm{PP}, c) \notin \{(\emptyset, \mathbf{0}), (\emptyset, \mathbf{1})\}$ and $(\mathrm{QQ}, d), (\mathrm{RR}, e) \neq (\emptyset, \mathbf{0})$. By the definition of $\oplus$ then also $(\mathrm{QQ}, d) \oplus (\mathrm{RR}, e) \neq (\emptyset, \mathbf{0})$ and again the simplified definition of $\otimes$ applies.

We first deal with the case $\mathrm{PP} \neq \emptyset$. By the above remarks, (17) and idempotence of $\oplus$ we obtain

$$(\mathrm{PP}, c) \otimes ((\mathrm{QQ}, d) \oplus (\mathrm{RR}, e)) \;=\; (\mathrm{PP}, c) \;=\; (\mathrm{PP}, c) \oplus (\mathrm{PP}, c) \;=\; (\mathrm{PP}, c) \otimes (\mathrm{QQ}, d) \oplus (\mathrm{PP}, c) \otimes (\mathrm{RR}, e) \ .$$

For the case $\mathrm{PP} \neq \emptyset$ we calculate

$$\begin{array}{ll} & (\mathrm{PP}, c) \otimes ((\mathrm{QQ}, d) \oplus (\mathrm{RR}, e)) \\ = & \quad \{\![\ \text{definition of } \oplus\ ]\!\} \\ & \begin{cases} (\mathrm{PP}, c) \otimes (\mathrm{QQ}, d) & \text{if } d > e \\ (\mathrm{PP}, c) \otimes (\mathrm{RR}, e) & \text{if } e > d \\ (\mathrm{PP}, c) \otimes (\mathrm{QQ} \cup \mathrm{RR}, d) & \text{if } d = e \end{cases} \\ = & \quad \{\![\ \text{above remarks and (17)}\ ]\!\} \\ & \begin{cases} (\mathrm{QQ}, c \cdot d) & \text{if } d > e \\ (\mathrm{RR}, c \cdot e) & \text{if } e > d \\ (\mathrm{QQ} \cup \mathrm{RR}, c \cdot d) & \text{if } d = e \end{cases} \\ = & \quad \{\![\ \text{cancellativity, (15) and } c \neq \mathbf{0}\ ]\!\} \\ & \begin{cases} (\mathrm{QQ}, c \cdot d) & \text{if } c \cdot d > c \cdot e \\ (\mathrm{RR}, c \cdot e) & \text{if } c \cdot e > c \cdot d \\ (\mathrm{QQ} \cup \mathrm{RR}, c \cdot d) & \text{if } c \cdot d = c \cdot e \end{cases} \\ = & \quad \{\![\ \text{definition of } \oplus\ ]\!\} \\ & (\mathrm{QQ}, c \cdot d) \oplus (\mathrm{RR}, c \cdot e) \\ = & \quad \{\![\ (17)\ ]\!\} \\ & (\mathrm{PP}, c) \otimes (\mathrm{QQ}, d) \;\oplus\; (\mathrm{PP}, c) \otimes (\mathrm{RR}, e) \ . \end{array}$$

Finally, it remains to show that the no detours rule holds. We do this by showing that the semiring is test-discrete and that $(\emptyset, \mathbf{1})$ is the greatest element. Assume that the semiring is not test-discrete. That means that there are elements $(\mathrm{PP}, c), (\mathrm{QQ}, d) \notin \{(\emptyset, \mathbf{0}), (\emptyset, \mathbf{1})\}$ such that $(\mathrm{PP}, c) \otimes (\mathrm{QQ}, d) = (\emptyset, \mathbf{0})$. By the definition of $\otimes$ this implies $c \cdot d = \mathbf{0}$, which, due to (16), only holds if $c = \mathbf{0}$ or $d = \mathbf{0}$. Hence, by (14), $(\mathrm{PP}, c) = (\emptyset, \mathbf{0})$ or $(\mathrm{QQ}, d) = (\emptyset, \mathbf{0})$. This contradicts our assumption. The equation $(\mathrm{PP}, c) \oplus (\emptyset, \mathbf{1}) = (\emptyset, \mathbf{1})$ follows by the definition of $\oplus$, Requirement (14) and the fact that S is a cost semiring. □