CALCULATING A PATH ALGORITHM

Roland C. Backhouse and A.J.M. van Gasteren

Department of Mathematics and Computing Science,
Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven,
The Netherlands.

Abstract. A calculational derivation is given of an abstract path algorithm, one instance of the algorithm being Dijkstra's shortest-path algorithm, another being breadth-first/depth-first search of a directed graph. The basis for the derivation is the algebra of regular languages.

1 Problem Statement

Given is a (non-empty) set N and a $|N| \times |N|$ matrix A, the rows and columns of which are indexed by elements of N. It is assumed that the matrix elements are drawn from a regular algebra[1] $(S, +, \cdot, *, 0, 1)$ having the two additional properties that

(1) the ordering $\leq$ induced[2] by $+$ is a total ordering on the elements of S, and

(2) 1 is the largest element in the ordering.

In addition to N and A, one is given a $1 \times |N|$ matrix b. Hereafter $1 \times |N|$ matrices will be called "vectors", 1×1 matrices will be called "elements" and $|N| \times |N|$ matrices will be called "matrices".

The problem is to derive an algorithm to compute the vector $b \cdot A*$ whereby the primitive terms in the algorithm do not involve the "$*$" operator.

2 Interpretations

The relevance and interest of the stated problem is that it is an abstraction from several path problems on labelled, directed graphs. Let $G = (N, E)$ be a directed graph with node set N and labelled-edge set E, the labels being drawn from some regular algebra $(S, +, \cdot, *, 0, 1)$. Then (as we deem known, see for instance [3] and [7],) there is a correspondence between edge sets E and $|N| \times |N|$ matrices A whereby the (i, j)th element of A equals the label of the edge from node i to node j in the graph, if present, and 0 otherwise. (For graphs with multiple edges, the correspondence will be adjusted later; note that absent edges and edges with label 0 are not distinguished.)

---

[1] Sometimes known as the algebra of regular languages. See [2] for the axiomatisation of regular algebra assumed here.

[2] $x \leq y \;\equiv\; x + y = y$

Since S forms a regular algebra $(S, +, \cdot, *, 0, 1)$, matrix multiplication can be defined in the usual way with the usual properties. In fact one can prove that the set of square matrices of a fixed size with entries drawn from S itself forms a regular algebra, with the usual definitions of the zero and identity matrices, matrix addition and matrix multiplication. The derivation to be presented makes extensive use of this fact. (For details see [3] .)

If the label of a path is defined to be the product of its constituent edge labels (taken in the order defined by the path), the (i,j)th entry of A^k is the sum of the labels of the paths of length k from node i to node j. Note that this still holds true if we redefine the (i,j)th entry of A to be the *sum* of the labels of all edges from i to j, thus admitting multiple edges. Moreover, if b is a vector that differs from the zero vector only in its ith entry, for some node i, then the jth entry of $b \cdot A*$ is the sum of all labels of paths beginning at node i and ending at node j.

Three interpretations of a regular algebra satisfying (1) and (2) are given in the table below. (Note that property (2) implies that $x* = 1$ for all $x \in S$. For this reason the interpretation of $*$ has been omitted.)

Table 1. Regular algebras

	S	$+$	$\cdot$	0	1	$\leq$
Shortest paths	nonnegative reals	$\downarrow$	$+$	∞	0	$\geq$
Reachability	booleans	$\vee$	$\wedge$	false	true	$\Rightarrow$
Bottlenecks	reals	$\uparrow$	$\downarrow$	$-\infty$	∞	$\leq$

The first interpretation is that appropriate to finding shortest paths; the length (i.e label) of a path is the (arithmetic) sum of its constituent edge lengths (labels) and for each node x the minimum (denoted in the table by "$\downarrow$") such length is sought to node x from a given, fixed start node. In this case, the algorithm we derive is known as Dijkstra's shortest path algorithm [5] (the carrier set, S, being restricted to *nonnegative* reals in order to comply with requirement (2)).

The second interpretation is that appropriate to solving reachability problems; the (i,j)th entry of matrix A is **true** if and only if there is an edge in E from node i to j, and the "length" of a path is always **true**. The algorithm we derive forms the basis of the so-called depth-first and breadth-first search methods for determining for each node in the graph whether or not there is a path in the graph to that node from the given start node. (The choice of "breadth-first" or "depth-first" search depends on further refinement steps that we do not discuss in detail.)

The third interpretation is appropriate to bottleneck problems; the edge labels may be construed as, say, bridge widths and the "length" of a path is the minimum width of bridge on that path. Sought is, for each node in the graph, the minimum bridge-width on a route to the node from a given start node that maximises that minimum. In this case the algorithm does not appear to have any specific name.

For more discussion of these interpretations see [3].

Another, somewhat simpler, application of the calculational techniques discussed here is a derivation of the Floyd/Warshall all-paths algorithm. For details see [1].

3 Selectors

Although the matrices we consider form a regular algebra, they will obviously never satisfy the requirements (1) and (2), even if their elements do. Given that we wish to appeal to these properties from time to time, it is important to keep track of which terms in our calculations denote elements, which denote vectors and which denote matrices. The rules for doing so are simple and (hopefully) familiar: the product of an $m \times n$ and an $n \times p$ matrix is an $m \times p$ matrix, and addition and $*$ preserve the dimension of their arguments. It remains, therefore, to adopt a systematic naming convention for the variables that we use. This, and a primitive mechanism for forming vectors, is the topic of this section.

During the course of the development the following naming conventions will be used.

$\mathbf{0}$	$1\times \mid N \mid$ vector that is everywhere 0.
k, j	nodes of the graph (i.e. elements of N)
L, M, P	subsets of N
V, W, X, Y, Z	$\mid N \mid \times \mid N \mid$ matrices
u, v, w, x	$1\times \mid N \mid$ vectors

For each node k we denote by $k\bullet$ the $1\times \mid N \mid$ vector that differs from $\mathbf{0}$ only in its kth component which is 1. Such a vector is called a *primitive selector vector*. The transpose of $k\bullet$ (thus a $\mid N \mid \times 1$ vector) is denoted by $\bullet k$. We define the *primitive selector matrix* $\underline{k}$ by the equation

(3) $\qquad \underline{k} \;=\; \bullet k \cdot k\bullet$

Any sum (including the empty sum, of course) of primitive selector vectors (respectively matrices) is called a *selector vector* (respectively *matrix*). In fact, in most cases, instead of using one of these four terms we shall just use the term *selector*, it being clear from the context whether the designated selector is primitive or not, and a vector or a matrix.

This terminology is motivated by the interpretation of the product of a matrix and a selector. Specifically, $j\bullet \cdot Y$ is a vector consisting of a copy of the jth row of matrix Y, and $j\bullet \cdot Y \cdot \bullet k$ is the (j, k)th element of Y. Furthermore, there is a (1-1) correspondence between subsets of N and selector matrices given by the function mapping M to $\underline{M}$ where, by definition,

(4) $\qquad \underline{M} \;=\; \Sigma(k : \; k \in M : \; \underline{k})$

(Note that

(5) $\qquad \underline{\{k\}} \;=\; \underline{k}$

This silent "lifting" of a function from elements to sets is not uncommon and very convenient; it should not be a cause for confusion since we do not mix the two forms, preferring always to use the shorter form.)

For all vectors x and all matrices Y, $x \cdot \underline{M}$ is a copy of x except that all elements of x with index outwith M are zero, and $Y \cdot \underline{M}$ (respectively, $\underline{M} \cdot Y$) is a copy of matrix Y except that all columns (respectively, rows) of Y with index outwith M are zero.

The derivation that follows is not dependent on knowing these interpretations of the selectors; rather, we make use of a small number of characteristic algebraic properties. The first is that matrix product is associative. This is the most important property and its exploitation is the reason for introducing the primitive selectors. However, we shall nowhere explicitly mention the use of associativity, in line with the doctrine that the application of the most important properties should be invisible. The second property is that $\underline{\phi}$ (where ϕ denotes the empty set of nodes) is the zero element in the algebra of $|N| \times |N|$ matrices. In particular, for all vectors u,

$$u \cdot \underline{\phi} \quad = \quad \mathbf{0} \tag{6}$$

and, for all matrices X,

$$X \cdot \underline{\phi} \quad = \quad \underline{\phi} \cdot X \quad = \quad \underline{\phi} \tag{7}$$

and

$$X + \underline{\phi} \quad = \quad X \tag{8}$$

(These properties are immediate consequences of the definition of the "underlining" function.) Two other properties are that, for all nodes k,

$$k\bullet \cdot \bullet k \quad = \quad 1 \tag{9}$$

and, for all distinct nodes j and k,

$$j\bullet \cdot \bullet k \quad = \quad 0 \tag{10}$$

It follows, by straightforward calculation, that, for all sets of nodes M and all nodes k,

$$k \in M \quad \Rightarrow \quad (\underline{M} \cdot \bullet k \;=\; \bullet k \quad \wedge \quad \underline{M} \cdot \underline{k} \;=\; \underline{k}) \tag{11}$$

and, for all sets of nodes L and M,

$$\underline{L} \cdot \underline{M} \quad = \quad \underline{L \cap M} \tag{12}$$

In particular,

$$\underline{M} \cdot \underline{M} \quad = \quad \underline{M} \tag{13}$$

and

$$L \cap M \;=\; \phi \quad \Rightarrow \quad \underline{L} \cdot \underline{M} \quad = \quad \underline{\phi} \tag{14}$$

The final property is that all matrices and all vectors are indexed by the given node set N. This we render by the equations:

$$X \cdot \underline{N} \quad = \quad X \quad = \quad \underline{N} \cdot X \tag{15}$$

and

$$x \cdot \underline{N} \quad = \quad x \tag{16}$$

for all matrices X and all vectors x.

4 The Key Theorem

A key insight in deriving an algorithm is that, because of properties (1) and (2), for any vector x and any matrix Y, at least one element of $x \cdot Y*$ is easy to compute. Specifically, choose k such that

$$(17) \qquad \forall(j : j \in N : x \cdot \bullet j \ \leq \ x \cdot \bullet k)$$

Thus the kth element of x is the largest. (Such a choice can always be made because the ordering on elements is total.) Then we claim that

$$(18) \qquad x \cdot Y * \cdot \bullet k \ = \ x \cdot \bullet k$$

In other words, no computation whatsoever is required to compute this one element.

Inevitably, the property we need in the development is more general; for instance, we shall have to comply with an additional condition $k \in M$, for some given M. As a consequence, we have to weaken (17) by strengthening its range to $j \in M$. We take the liberty of immediately stating and establishing the required generalisation, because its proof is virtually the same.

Theorem 19 For all $M \subseteq N$, $k \in N$, matrices Z and vectors u,

$$\begin{array}{l} k \in M \ \wedge \ \forall(j : j \in M : u \cdot \bullet j \ \leq \ u \cdot \bullet k) \ \wedge \ Z \geq \underline{N} \\ \Rightarrow \quad u \cdot \underline{M} \cdot Z \cdot \bullet k \ = \ u \cdot \bullet k \end{array}$$

Proof The proof is by mutual inclusion. Assume the antecedent of the implication. Then,

$$\begin{array}{ll} & u \cdot \underline{M} \cdot Z \cdot \bullet k \\ = & \quad \{ \text{ definition of } \underline{M} \ \} \\ & u \cdot \Sigma(j : j \in M : \bullet j \cdot j\bullet) \cdot Z \cdot \bullet k \\ = & \quad \{ \text{ distributivity } \} \\ & \Sigma(j : j \in M : u \cdot \bullet j \cdot j\bullet \cdot Z \cdot \bullet k) \\ \leq & \quad \{ \text{ properties of } k\text{, monotonicity } \} \\ & \Sigma(j : j \in M : u \cdot \bullet k \cdot j\bullet \cdot Z \cdot \bullet k) \\ \leq & \quad \{ \ j\bullet \cdot Z \cdot \bullet k \text{ is an element. Hence,} \\ & \quad\ \ j\bullet \cdot Z \cdot \bullet k \ \leq \{ \text{ (2) } \} \ 1 \ , \quad \text{monotonicity } \} \\ & \Sigma(j : j \in M : u \cdot \bullet k \cdot 1) \\ \leq & \quad \{ \text{ calculus } \} \\ & u \cdot \bullet k \end{array}$$

and,

$$\begin{array}{ll} & u \cdot \underline{M} \cdot Z \cdot \bullet k \\ \geq & \quad \{ \text{ assumption: } Z \geq \underline{N} \ \} \\ & u \cdot \underline{M} \cdot \underline{N} \cdot \bullet k \\ = & \quad \{ \text{ (15) } \} \\ & u \cdot \underline{M} \cdot \bullet k \\ = & \quad \{ \text{ assumption: } k \in M, \ (11) \ \} \\ & u \cdot \bullet k \end{array}$$

□

The reader is invited to check that property (18) does indeed follow from (17) by application of the theorem: with u instantiated to x, M instantiated to N and Z to $Y*$ the condition $k \in M$ is automatically satisfied and $Z \geq \underline{N}$ is satisfied by virtue of the fact that $Y* \geq \underline{N}$ for all $|N| \times |N|$ matrices Y. (The latter property follows since, by definition, $Y* \ = \ \underline{N} \ + \ Y* \cdot Y$.)

5 A Skeleton Algorithm

The algorithm we develop is based on an iterative process in which at each iteration theorem 19 is used to "eliminate" one node from the calculation. At some intermediate stage some set of nodes has been eliminated and some set M of nodes remains to be considered. These heuristics are captured formally by postulating as loop invariant

(20) $\quad b \cdot A* \ = \ x \cdot \underline{M} \cdot Y* \ + \ w$

and as postcondition

(21) $\quad b \cdot A* \ = \ w$

where x and w are vectors and Y is a matrix. (In the course of our calculations we shall be obliged to strengthen this invariant.)

Using elementary properties of regular algebra together with the property (15) of the selector matrix $\underline{N}$ it is easy to see that the invariant is established by the assignment

(22) $\quad x, Y, M, w \ := \ b, A, N, \mathbf{0}$

For the guard of the loop there is an abundance of choice. We shall therefore "hedge our bets" for the moment by choosing the weakest possible guard, namely the negation of the postcondition (21). Note, however, that the postcondition is implied by (although not equivalent to) the conjunction of $M = \phi$ and the loop invariant. Since N is a finite set, progress towards $M = \phi$ (and thus towards termination) is guaranteed if at each iteration one node is removed from M. The key theorem discussed in the last section provides a mechanism for choosing such a node. This is the main reason why invariant (20) is preferable to an otherwise equally reasonable invariant like

$$b \cdot A* \ = x \cdot Y* \ + \ w$$

Thus, the main design decision for the loop body is to choose k, $k \in M$, in such a way that

(23) $\quad x \cdot \underline{M} \cdot Y* \cdot \underline{k} \ = \ x \cdot \underline{k}$

and to try to "transfer" $x \cdot \underline{k}$ from $x \cdot \underline{M} \cdot Y*$ to w. The key theorem, indeed, enables this choice since, by multiplying both sides of its consequent by $k\bullet$ and simplifying $\bullet k \cdot k \bullet$ to $\underline{k}$, we obtain the lemma: for all vectors u and matrices Z,

(24) $\quad Z \ \geq \ \underline{N} \ \Rightarrow \ u \cdot \underline{M} \cdot Z \cdot \underline{k} \ = \ u \cdot \underline{k}$

This leads to the following skeleton algorithm. (For greater clarity the different components of the simultaneous assignment to M, w, x and Y have been separated by "$\|$" symbols.)

$$
\begin{array}{llll}
 & x, Y, M, w \;:=\; b, A, N, \mathbf{0} & & \\
; & \{ \text{ Invariant: } & b \cdot A* \;=\; x \cdot \underline{M} \cdot Y* \;+\; w & \\
 & & \text{Variant: } \; |M| \} & \\
\texttt{do} & b \cdot A* \;\neq\; w \;\longrightarrow & & \\
 & & \texttt{choose} \;\; k \in M \text{ s.t. } \forall (j :\; j \in M :\; x \cdot \bullet j \;\leq\; x \cdot \bullet k) & \\
; & & \quad M \;:=\; M - \{k\} & \\
 & & \| \;\; w \;:=\; w' & \\
 & & \| \;\; x, Y \;:=\; x', Y' & \\
\texttt{od} & & & \\
\{ \; b \cdot A* \;=\; w \} & & &
\end{array}
$$

For convenience we let P denote $M - \{k\}$; i.e.

(25) $\quad \underline{M} \;=\; \underline{P} \;+\; \underline{k}$

Then, by a standard use of the assignment axiom, we obtain a requirement on the three unknowns, namely:

(26) $\quad x \cdot \underline{M} \cdot Y* \;+\; w \;=\; x' \cdot \underline{P} \cdot (Y')* \;+\; w'$

Before embarking on the calculation dealing with (26) it may be helpful to summarise the properties of regular algebra that are useful for dealing with expressions $Z*$. For all matrices Z the defining equation for $Z*$ is

(27) $\quad Z* \;=\; \underline{N} \;+\; Z* \cdot Z$

and, hence,

(28) $\quad Z* \;\geq\; \underline{N}$

Finally, two general properties of a regular algebra are the rule we call "the leapfrog rule": for all X and Y,

(29) $\quad V \cdot (W \cdot V)* \;=\; (V \cdot W)* \cdot V$

and the rule we call "star decomposition":

(30) $\quad (V{+}W)* \;=\; (V* \cdot W)* \cdot V*$

6 The Assignment to w

We begin our calculation by considering the transfer of $x \cdot \underline{k}$ from $x \cdot \underline{M} \cdot Y*$ to w. That is to say, inspired by (23) and (25), we propose choosing $w' \;=\; x \cdot \underline{k} \;+\; w$ and seek a suitable term Z such that

(31) $\quad x \cdot \underline{M} \cdot Y* \;=\; Z \;+\; x \cdot \underline{k}$

We have:

$$
\begin{array}{ll}
 & x \cdot \underline{M} \cdot Y* \\
= & \quad \{ \ \underline{M} = \underline{M} \cdot \underline{M} \\
 & \quad \bullet \ \text{assume } Y = Y \cdot \underline{M} \\
 & \quad \ \ \text{in preparation for the use of the leapfrog rule } \} \\
 & x \cdot \underline{M} \cdot \underline{M} \cdot (Y \cdot \underline{M})* \\
= & \quad \{ \ \text{leapfrog rule } \} \\
 & x \cdot \underline{M} \cdot (\underline{M} \cdot Y)* \cdot \underline{M} \\
= & \quad \{ \ \underline{M} = \underline{P} + \underline{k} \ \} \\
 & x \cdot \underline{M} \cdot (\underline{M} \cdot Y)* \cdot (\underline{P} + \underline{k}) \\
= & \quad \{ \ \text{distributivity } \} \\
 & x \cdot \underline{M} \cdot (\underline{M} \cdot Y)* \cdot \underline{P} + x \cdot \underline{M} \cdot (\underline{M} \cdot Y)* \cdot \underline{k} \\
= & \quad \{ \ \text{choice of } k, \text{(28) and (24) } \} \\
 & x \cdot \underline{M} \cdot (\underline{M} \cdot Y)* \cdot \underline{P} + x \cdot \underline{k}
\end{array}
$$

In this calculation the first step is the most crucial. The goal is to reap as much advantage as possible from the key property (24). Knowing that "$\underline{k}$" can always be introduced into our calculations by the identity $\underline{M} = \underline{P} + \underline{k}$, the strategy is to retain "$x \cdot \underline{M}$" whilst simultaneously introducing a second occurrence of "$\underline{M}$" on the right of the expression being manipulated. (Quotation marks are used in this discussion because the considerations are entirely syntactic.) To do this an extra property of the matrix Y is required — as indicated by the bullet in the first hint. In order to accommodate this extra requirement we strengthen the invariant to:

(32) $\qquad b \cdot A* = x \cdot \underline{M} \cdot (Y \cdot \underline{M})* + w$

whereby Y has been replaced by $Y \cdot \underline{M}$. Incorporating the assignment to w and the new invariant we arrive at the following refinement of our initial skeleton algorithm.

$$
\begin{array}{lll}
 & x, Y, M, w := b, A, N, \mathbf{0} & \\
; & \{ \text{Invariant: } & b \cdot A* = x \cdot \underline{M} \cdot (Y \cdot \underline{M})* + w \\
 & & \text{Variant: } |M|\} \\
 & \texttt{do} \quad b \cdot A* \neq w \longrightarrow & \\
 & & \texttt{choose} \ \ k \in M \text{ s.t. } \forall(j : j \in M : x \cdot \bullet j \leq x \cdot \bullet k) \\
 & ; & \quad M := P \ \texttt{where} \ P = M - \{k\} \\
 & & \| \ w := x \cdot \underline{k} + w \\
 & & \| \ x, Y := x', Y' \\
 & \texttt{od} & \\
\{ & b \cdot A* = w \} &
\end{array}
$$

Strengthening the invariant demands that we check that the initialisation is still correct — which it is on account of (15) — and revise the requirement on x' and Y'. Applying the assignment axiom once again, maintenance of the invariant is guaranteed by the identity

(33) $\qquad x \cdot \underline{M} \cdot (Y \cdot \underline{M})* + w = x' \cdot \underline{P} \cdot (Y' \cdot \underline{P})* + x \cdot \underline{k} + w$

However, by the calculation above,

$$
\begin{array}{ll}
 & x \cdot \underline{M} \cdot (Y \cdot \underline{M})* \; + \; w \\
= & \\
 & x \cdot \underline{M} \cdot (\underline{M} \cdot Y)* \cdot \underline{P} + x \cdot \underline{k} + w
\end{array}
$$

Hence, it suffices to choose x' and Y' so that

(34) $\qquad x \cdot \underline{M} \cdot (\underline{M} \cdot Y) * \cdot \underline{P} \;=\; x' \cdot \underline{P} \cdot (Y' \cdot \underline{P})*$

7 Using Star Decomposition

One half of our calculation is now complete. In the remaining half we deal with the assignments to x and Y. Our first tactic is to get into a position to invoke the star-decomposition rule. Considering the left side of (34) but omitting the final term in the product, we have:

$$
\begin{array}{ll}
 & x \cdot \underline{M} \cdot (\underline{M} \cdot Y)* \\
= & \quad \{ \; \underline{M} = \underline{P} + \underline{k}, \text{distributivity} \; \} \\
 & x \cdot \underline{M} \cdot (\underline{P} \cdot Y + \underline{k} \cdot Y)*
\end{array}
$$

The desired position has been reached. We now aim to use the choice of k — more precisely, (24) — with Z instantiated to as large a value as possible. This involves exposing "$\underline{k}$" outwith the "*" operator. For brevity and clarity introduce the temporary abbreviations V for $\underline{P} \cdot Y$ and W for $\underline{k} \cdot Y$. Then we proceed as follows.

$$
\begin{array}{ll}
 & x \cdot \underline{M} \cdot (\underline{P} \cdot Y + \underline{k} \cdot Y)* \\
= & \quad \{ \; \text{definitions of } V \text{ and } W \; \} \\
 & x \cdot \underline{M} \cdot (V + W)* \\
= & \quad \{ \; \text{consider the term } (V{+}W)*. \text{ Recalling that } W \;=\; \underline{k} \cdot Y, \\
 & \qquad \text{we try to derive an equal expression in which } W \text{ is} \\
 & \qquad \text{exposed as far to the right as possible.} \\
 & \qquad \begin{array}{ll}
 & (V + W)* \\
 = & \quad \{ \; \text{star decomposition} \; \} \\
 & (V* \cdot W)* \cdot V* \\
 = & \quad \{ \; (27) \; \} \\
 & (\underline{N} + (V* \cdot W)* \cdot V* \cdot W) \cdot V* \\
 = & \quad \{ \; \text{star decomposition} \; \} \\
 & (\underline{N} + (V + W)* \cdot W) \cdot V*
 \end{array} \\
 & \quad \} \\
 & x \cdot \underline{M} \cdot (\underline{N} + (V + W)* \cdot W) \cdot V* \\
= & \quad \{ \; \text{distributivity, definition of } W \; \} \\
 & (x \cdot \underline{M} \cdot \underline{N} + x \cdot \underline{M} \cdot (V + W)* \cdot \underline{k} \cdot Y) \cdot V* \\
= & \quad \{ \; (15), \text{and } (24) \text{ using } (V{+}W)* \geq \; \{ \; (28) \; \} \; \underline{N} \; \} \\
 & (x \cdot \underline{M} + x \cdot \underline{k} \cdot Y) \cdot V* \\
= & \quad \{ \; \text{definition of } V \; \} \\
 & (x \cdot \underline{M} + x \cdot \underline{k} \cdot Y) \cdot (\underline{P} \cdot Y)*
\end{array}
$$

Multiplying both sides of the obtained equality by the omitted term $\underline{P}$ and continuing we get:

$$
\begin{array}{ll}
 & x \cdot \underline{M} \cdot (\underline{M} \cdot Y)* \cdot \underline{P} \\
= & \qquad \{ \text{ above calculation } \} \\
 & (x \cdot \underline{M} + x \cdot \underline{k} \cdot Y) \cdot (\underline{P} \cdot Y)* \cdot \underline{P} \\
= & \qquad \{ \text{ leapfrog rule } \} \\
 & (x \cdot \underline{M} + x \cdot \underline{k} \cdot Y) \cdot \underline{P} \cdot (Y \cdot \underline{P})* \\
= & \qquad \{ \text{ distributivity; and } P \subseteq M, \\
 & \qquad\quad \text{thus } x \cdot \underline{M} \cdot \underline{P} \ = \ \{ \ (12) \ \} \ x \cdot \underline{P} \\
 & \qquad \} \\
 & (x + x \cdot \underline{k} \cdot Y) \cdot \underline{P} \cdot (Y \cdot \underline{P})*
\end{array}
$$

Our calculation is now complete. Comparing the last line of the above calculation with the stated requirements on x' and Y' (see equation (34)), one sees that $Y = Y'$ and $x' \cdot \underline{P} = (x + x \cdot \underline{k} \cdot Y) \cdot \underline{P}$ suffice. Hence possible choices for x' are

$$x + x \cdot \underline{k} \cdot Y$$

and, since $\underline{P} \cdot \underline{P} = \underline{P}$,

$$(x + x \cdot \underline{k} \cdot Y) \cdot \underline{P}$$

We choose the latter for reasons discussed in the next section, the main reason being that this choice, by construction, yields

$$x \cdot \underline{M} = x$$

as an additional invariant. (It holds initially, i.e. for $M = N$.) Thus, apart from simplification of the termination condition, the complete algorithm is as follows:

$$
\begin{array}{lll}
\multicolumn{3}{l}{x, Y, M, w \ := \ b, A, N, \mathbf{0}} \\
; & \{ \text{Invariant: } & b \cdot A* \ = \ x \cdot \underline{M} \cdot (Y \cdot \underline{M})* \ + \ w \\
 & & \text{Variant: } \ |M|\} \\
\texttt{do} & b \cdot A* \ \neq \ w \ \longrightarrow & \\
 & & \texttt{choose} \ \ k \in M \text{ s.t. } \forall(j : \ j \in M : \ x \cdot \bullet j \ \leq \ x \cdot \bullet k) \\
 \quad ; & & \quad M \ := \ P \\
 & & \| \ w \ := \ x \cdot \underline{k} \ + \ w \\
 & & \| \ x \ := \ (x + x \cdot \underline{k} \cdot Y) \ \cdot \ \underline{P} \\
 & & \quad \texttt{where} \ P \ = \ M - \{k\} \\
\texttt{od} & & \\
\multicolumn{3}{l}{\{ \ b \cdot A* \ = \ w \}}
\end{array}
$$

8 Elementwise Implementation

In this final section we make one last significant modification to the invariant, we indicate how to reexpress the vector assignments in terms of elementwise operations and we at long last make a decision on the termination condition. Some remarks on the relationship between the algorithm presented here and conventional descriptions of Dijkstra's shortest-path algorithm and of traversal algorithms are also included.

A simple, but nevertheless significant, observation is that, with L defined by

$$(35) \qquad L \;=\; N - M$$

the assignments to w establish and subsequently maintain invariant the property:

$$(36) \qquad w \;=\; w \cdot \underline{L}$$

(The set L would conventionally be called the "black" nodes; see, for example, Dijkstra and Feijen's account [6] of Dijkstra's shortest-path algorithm.) Exploitation of this property was anticipated when we expressed our preferred choice of assignment to x. For, as stated in the previous section, the assignments to x establish and maintain invariant the property

$$(37) \qquad x \;=\; x \cdot \underline{M}$$

Now, suppose we introduce the vector u where, by definition,

$$(38) \qquad u \;=\; x + w$$

Then, since $L \cap M \;=\; \phi$, we have by (13) and (14)

$$(39) \qquad x \;=\; u \cdot \underline{M}$$

and

$$(40) \qquad w \;=\; u \cdot \underline{L}$$

In other words, there is a (1-1) correspondence between the vector u and the pair of vectors (x, w). We are thus free to replace this pair in the algorithm by u. This we do as follows.

In the invariant we use (37), (39) and (40) to remove occurrences of x and w; we obtain:

$$(41) \qquad b \cdot A* \;=\; u \cdot \underline{M} \cdot (Y \cdot \underline{M})* \;+\; u \cdot \underline{L}$$

In the statement 'choose k ...', in the body of the loop, we replace occurrences of x by u; this is allowed since by (39) and (11) we have, for $j \in M$,

$$x \cdot \bullet j \;=\; u \cdot \bullet j$$

For the assignments to u we simply add the right sides of the assignments to x and w (cf. (38)) and again use (39) and (40) to remove all traces of these variables. We have

$$\begin{array}{ll}
 & (x + x \cdot \underline{k} \cdot Y) \;\cdot\; \underline{P} \;+\; x \cdot \underline{k} \;+\; w \\
= & \qquad \{ \text{ distributivity } \} \\
 & x \cdot \underline{P} + x \cdot \underline{k} \cdot Y \cdot \underline{P} + x \cdot \underline{k} + w \\
= & \qquad \{ \text{ commutativity of } + \; \} \\
 & x \cdot \underline{P} + x \cdot \underline{k} + w + x \cdot \underline{k} \cdot Y \cdot \underline{P} \\
= & \qquad \{ \text{ distributivity, } \underline{M} \;=\; \underline{P} + \underline{k} \; \}
\end{array}$$

$$
\begin{array}{ll}
& x \cdot \underline{M} + w + x \cdot \underline{k} \cdot Y \cdot \underline{P} \\
= & \qquad \{ \ (37), (38) \text{ and } (39) \ \} \\
& u + u \cdot \underline{M} \cdot \underline{k} \cdot Y \cdot \underline{P} \\
= & \qquad \{ \ k \in M, (11) \ \} \\
& u + u \cdot \underline{k} \cdot Y \cdot \underline{P}
\end{array}
$$

Thus, in the body of the loop the assignment to u is

(42) $\qquad u \ := \ u \ + \ u \cdot \underline{k} \cdot Y \cdot \underline{P}$

What this entails at element level can be ascertained by postmultiplying by $\bullet j$, for each node j. For $j \in P$, we calculate as follows:

$$
\begin{array}{ll}
& (u + u \cdot \underline{k} \cdot Y \cdot \underline{P}) \ \cdot \ \bullet j \\
= & \qquad \{ \ \text{distributivity} \ \} \\
& u \ \cdot \ \bullet j \ + \ u \ \cdot \ \underline{k} \ \cdot \ Y \ \cdot \ \underline{P} \ \cdot \ \bullet j \\
= & \qquad \{ \ \text{definition of } \underline{k}, (5); \text{ assumption } j \in P \text{ and } (11) \ \} \\
& (u \ \cdot \ \bullet j) \ + \ (u \ \cdot \ \bullet k) \ \cdot \ (k \bullet \ \cdot \ Y \ \cdot \ \bullet j)
\end{array}
$$

Parentheses have been included in the last line in order to indicate which subexpressions are elements. For $j \notin P$ the second summand —in the penultimate formula of the calculation— reduces to zero, since $\underline{P} \cdot \bullet j \ = \ 0$. I.e.

(43) $\qquad (u + u \cdot \underline{k} \cdot Y \cdot \underline{P}) \ \cdot \ \bullet j \ \ = \ \ u \cdot \bullet j$

The final decision we have to make is on the termination condition. For our own objectives here we make do with the simplest possible termination condition, namely $M \ = \ \phi$, thus obtaining the following algorithm:

$$
\begin{array}{l}
\quad u, Y, M \ := \ b, A, N \\
; \qquad \{ \text{ Invariant: } \ b \cdot A* \ \ = \ \ u \cdot \underline{M} \cdot (Y \cdot \underline{M})* \ + \ u \cdot \underline{N - M} \\
\qquad\quad \text{ Variant: } \ |M| \} \\
\quad \texttt{do} \ \ M \neq \phi \ \ \longrightarrow \\
\qquad\quad \texttt{choose} \ \ k \in M \text{ s.t. } \forall (j : \ j \in M : \ u \cdot \bullet j \ \leq \ u \cdot \bullet k) \\
\qquad ; \qquad M \ := \ P \\
\qquad\quad \| \ \texttt{parfor} \ \ \ j \in P \ \ \texttt{do} \\
\qquad\qquad\quad u \cdot \bullet j \ := \ u \cdot \bullet j \ + \ (u \cdot \bullet k) \cdot (k \bullet \cdot Y \cdot \bullet j) \\
\qquad\qquad \texttt{where} \ P \ = \ M - \{k\} \\
\quad \texttt{od} \\
\{ \ b \cdot A* \ \ = \ \ u \}
\end{array}
$$

Note that, since $k \notin P$, the parallel `for` statement can be replaced by a sequential `for` statement.

As for other termination conditions, we remarked earlier that the set L corresponds to the so-called "black" nodes in Dijkstra and Feijen's description of Dijkstra's shortest-path algorithm. They also distinguish "white" and "grey" nodes. The "grey" nodes are just nodes $j \in M$ for which $u \cdot \bullet j \neq 0$, and the "white" nodes are the remaining nodes in M; in some circumstances (e.g. if the graph is not connected) advantage can be made from the special multiplicative and additive properties of the zero element of the algebra. In this case, a suitable choice for the termination condition would be $u \cdot \underline{M} = \mathbf{0}$: then $u \cdot \underline{N - M} = u$ and, hence, $b \cdot A* = \mathbf{0} \cdot (Y \cdot \underline{M})* + u = \mathbf{0} + u = u$. (In conventional terminology, this termination condition reads "the set of grey nodes is empty"; conventionally, this is, indeed, the choice made.)

A termination condition $u \cdot \underline{M} = \mathbf{0}$ requires the algorithm to keep track of the nodes $j \in M$ for which $u \cdot \bullet j \neq 0$. We won't go into the precise details of such an addition to the algorithm, but we finish by noting that, in the interpretation pertaining to reachability problems (see the table in section 2), keeping track of the "grey" nodes in a queue leads to a breadth-first traversal algorithm; using a stack instead of a queue leads to depth-first traversal.

9 Commentary and Credits

The goal of this report has been to show how a class of standard path algorithms can be derived by algebraic calculation. This is, of course, not the first and nor (we hope) will it be the last such derivation. The algebraic basis for the calculation given here was laid in [2], and some of its details were influenced by Carré 's derivation [3] of the same algorithm. A great many other authors have described and applied related algebraic systems to a variety of programming problems; Tarjan's paper [7] includes many references. The main distinguishing feature of the development presented here, however, is its reliance on calculations with matrices rather than with matrix elements, resulting (in our view) in a pleasingly compact presentation.

References

1. R.C. Backhouse. Calculating the Floyd/Warshall path algorithm. Eindhoven University of Technology, Department of Computing Science, 1992.
2. R.C. Backhouse and B.A. Carré. Regular algebra applied to path-finding problems. *Journal of the Institute of Mathematics and its Applications*, 15:161–186, 1975.
3. B.A. Carré. *Graphs and Networks.* Oxford University Press, 1979.
4. P. Chisholm. Calculation by computer. In *Third International Workshop Software Engineering and its Applications*, pages 713–728, Toulouse, France, December 3-7 1990. EC2.
5. E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
6. E.W. Dijkstra and W.H.J. Feijen. *Een Methode van Programmeren.* Academic Service, Den Haag, 1984. Also available as *A Method of Programming*, Addison-Wesley, Reading, Mass., 1988.
7. R.E. Tarjan. A unified approach to path problems. *Journal of the Association for Computing Machinery*, 28:577–593, 1981.

Acknowledgements

Preparation of this report was expedited by the use of the proof editor developed by Paul Chisholm [4].
This article was processed using the LaTeX macro package with LLNCS style