Dresden University of Technology
Department of Computer Science
Institute for Theoretical Computer Science

Algebraic Structures for Transitive Closure

Rafael Penaloza

Seminar “Graph Algorithms”
Prof. Dr.-Ing. habil. Heiko Vogler
Dr.-Ing. Armin Kühnemann

Contents

1 Introduction

Many real problems are well known to be representable by means of graphs. For these representations, algorithms have been developed that solve particular problems. Some examples of this are Warshall's algorithm for the reachability problem, or the Floyd-Warshall algorithm for computing the all-pair shortest path in a weighted graph.

An inspection on these algorithms shows that some of them follow a similar processing scheme, although applied to different mathematical structures. Noticing this similarity is the first step towards a generalization; that is, an algorithm that effectively solves the problems, and is not dependant on the specific domain over which they are defined.

This kind of generalization is desired for many reasons. One simple motivation is given by the fact that sometimes many questions may arise for the same graph. If a general algorithm is known to solve all those questions, then it is the only method that should be implemented, instead of making many implementations of question-specific algorithms. The particular case presented here will have the further property that a general solution may be obtained. This means that not only there is just one algorithm, but it needs only to be run once to obtain answers to all the questions.

Another interesting property of a generalization is that it may be applied to other problems that are currently not known. This means that, for solving a new problem, an algorithm is already known, and no effort is needed in order to find correct solving methods.

In order to achieve the generalization, one must first detect the elements that appear in each of the different known solutions, and try to find a general method of representing and processing them. In this case, an algebraic structure, the closed semiring, is presented as a structure that will be useful defining the desired general algorithm.

This algorithm will simply calculate the closure of a matrix of elements of a closed semiring. It will be shown that this closure will represent the solution, or a variation of the solution, of problems defined over graphs.

A very nice consequence of the present generalization is the ability to calculate inverses of matrices, by means of the same algorithm, as will be presented in the last section of this work.

This work is structured as follows. Section 2 presents a motivation for the search of a general algorithm. The section following it introduces the basic algebraic structure needed for achieving this task: the closed semiring. It also shows how to work on matrices over elements of this kind of algebra. At the end of the section, several examples of closed semirings are given.

On Section 4, the desired general algorithm is presented, which is applied in Section 5 to solve the examples presented in the motivation. At the end of that section, the generalization is used to solve a new problem not presented before.

2 Motivation

Suppose you are given a graph, for example the graph shown in Figure 1, and asked to find which nodes are reachable from which others. This is a very common problem, with a well-known solution. You need only to apply Warshall's algorithm to the adjacency matrix associated to the graph, and obtain the reflexive and transitive closure of this relation; afterwards, every entry in the closure matrix containing a one will refer to one of the desired pairs; in other words, the closure matrix shows the Boolean relation "reachable" between ordered pairs of nodes.

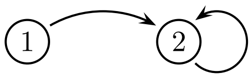

Fig. 1: A simple graph

The adjacency matrix in this particular case is

$$\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix},$$

and the application of Warshall's algorithm yields the expected result

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Now suppose that once you got this solution to the problem, someone else comes in and tells you that the graph you were seeing was incomplete, and should include some weights on its edges (see Figure 2). Furthermore, you are now asked to find out the shortest path between each pair of vertices in the graph, given that the weights in the graph represent a certain measure of distance between the nodes.

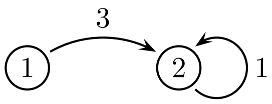

Fig. 2: The same graph, with some added weights

Once again, it is an easy problem, and another well-known algorithm (Floyd-Warshall algorithm) gives the desired answer. But using this algorithm means wasting one fore-calculated solution, and spending some time calculating a new one. In particular, it is necessary to construct a new matrix that describes the graph

$$\begin{pmatrix} \infty & 3 \\ \infty & 1 \end{pmatrix},$$

and then calculating the new solution, which, in this case, is

$$\begin{pmatrix} 0 & 3 \\ \infty & 0 \end{pmatrix}.$$

If instead of needing to calculate new solutions, there were a method that provides a general solution, this second calculation could have been avoided.

Furthermore, suppose now that a third person appears in the scene and tells you that you have been tricked, and the graph really represents an automaton (see Figure 3), with the 'weights' being the symbols in the alphabet by which the automaton changes its state.

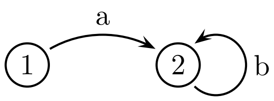

Fig. 3: An automaton

You are now asked to compute regular expressions for the accepted languages of this automaton for each of the possible ways to choose one initial and one final state. Once again, you may use Kleene's algorithm to fulfill this task, but you are getting bored of all these calculation, and wonder if this time your effort will be useful, or if you will be asked to change the methods once again. Even more, you want to make sure that this never happens again.

What you need is a general method, that will allow you to compute all the solutions at once, but without having to invest much additional time on the task. Here, a single algorithm capable of computing all the distinct solutions will be presented, with the added advantage that it allows also to calculate all of them in one single run. In fact, the algorithm was already presented in the examples above, but still requires some definitions and translations to make it useful.

3 Closed semirings

As the main goal is to generalize algorithms defined over matrices, it is first necessary to introduce general operations over these mathematical entities. The basic algebraic structure for working with matrices is a semiring, as two operations, addition and multiplication, are necessary to define operations over matrices.

A closed semiring is a structure that fulfills almost all the axioms of a usual semiring, and some additional ones. All the properties stated in the next definition will be used on the general algorithm in Section 4, and they are sufficient for solving the desired task.

Definition 3.1 *An algebraic structure* $C = (S, +, \cdot, ^*, 0, 1)$ *is called* ***closed semiring*** *if and only if it fulfills the following conditions:*

1. *S is a set; $+$ and $\cdot$ are two binary operators called* ***addition*** *and* ***multiplication****, respectively; * is a unary operator called* ***closure****; and 0 and 1 are elements of S*
2. *For every $a, b, c \in S, a + (b + c) = (a + b) + c$; addition is associative.*
3. *For every $a, b \in S, a + b = b + a$; addition is commutative.*
4. *For every $a \in S, a + 0 = a$; that is, 0 is a neutral element for addition.*
5. *For every $a, b, c \in S, a \cdot (b \cdot c) = (a \cdot b) \cdot c$; multiplication is associative.*
6. *For every $a \in S, a \cdot 1 = 1 \cdot a = a$; that is, 1 is a neutral element for multiplication.*
7. *For every $a, b, c \in S, a \cdot (b + c) = a \cdot b + a \cdot c$, and*

   $(a + b) \cdot c = a \cdot c + b \cdot c$; multiplication distributes over addition.
8. *For every $a \in S, a^* = 1 + a \cdot a^* = 1 + a^* \cdot a$.* $\square$

Note that this definition does not force the element 0 to be absorbing with respect to multiplication ($a \cdot 0 = 0 \cdot a = 0$). As the goal is to use matrices related to a given graph, the lack of this assumption yields to a limitation on the graphs, for they, in general must be complete in order to fully define a matrix. A graph is complete if for every pair of nodes in it there is an edge with an associated weight on it. In the case that 0 is also absorbing, missing edges are assumed to be edges with a weight equal to 0, but when it is not the case, no reasonable assumption may be made over missing edges.

Properties 1 to 7 are very standard, and it is not difficult to find algebraic structures that fulfill them. Nonetheless, many of those structures do not, in general, satisfy property 8. Example 3.7 shows one of such structures.

In general, it might be the case that the closure is not an operator, but a partial function from the carrier set to itself. In other words, the closure might be defined only for some, possibly none, of the elements in the set. In that case, the algebraic structure $(S, +, \cdot, ^*, 0, 1)$ may be completed into the semiring $\left(S \cup \{u\}, +', \cdot', ^{*'}, 0, 1\right)$ where $u \notin S$ and the new operators are just extensions of the original ones, by adding the following definitions for every $a \in S$:

$$u +' a = a +' u = u,$$

$$u \cdot' a = a \cdot' u = u,$$

$$a^{*'} = \begin{cases} u & \text{if } a \notin Dom\,(^*) \\ a^* & \text{if } a \in Dom\,(^*) . \end{cases}$$

The completion of a structure to a closed semiring forces the neutral element for addition to be non-absorbing with respect to multiplication, because $u \cdot 0 = u$. This is the main reason for not asking an absorbing 0 in the structure. As will be shown later, some interesting applications require a completion to be used.

Now, the operations for matrices over a closed semiring may be defined in the usual way.

Notation 3.1 *A matrix A of size $m \times n$ may be denoted by showing the element ocurring at every position, as follows*

$$A = [a_{ij}]_{1 \le i \le m, 1 \le j \le n} .$$

Definition 3.2 *Let A and B be matrices of size $n \times n$ over a closed semiring S. Then,*

$$A + B = [a_{ij} + b_{ij}]_{1 \le i,j \le n} ,$$
$$A \cdot B = \left[\sum_{k=1}^{n} a_{ik} \cdot b_{kj} \right]_{1 \le i,j \le n} .$$

□

The next two constant matrices will be useful for the following definitions, and to explain the closure of matrices.

$$O_n = [0]_{1 \le i,j \le n} ,$$
$$I_n = [\delta_{ij}]_{1 \le i,j \le n} , \quad \text{where } \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j. \end{cases}$$

Simple calculations show that the analog of properties 1 to 5, and 7 from Definition 3.1 are satisfied by matrices of size $n \times n$ over a closed semiring, with O_n and I_n working as neutral elements for addition and multiplication, respectively. For this, the closure is not taken into account, as it still remains to be defined.

The analogue of property 6, that is $A \cdot I_n = I_n \cdot A = A$, does not hold in general. This is again because 0 is not necessarily an absorbing element for multiplication, and may be the case that the addition included in the product of matrices includes a non-zero element apart from the desired one. It is strongly recomended to verify that this happens over the closed semiring G, that will be defined later in this section. This means that the matrices over a closed semiring do not define a closed semiring.

Not everything is lost, though, for a slightly weaker, but very useful, property is satisfied, as shown in the next lemma.

Lemma 3.3 *Let A and B be matrices of size $n \times n$ over a closed semiring, then*

1. $(I_n + A) \cdot B = B + A \cdot B$

2. $A \cdot (I_n + B) = A + A \cdot B$

Proof.

$$\begin{aligned}
[(I_n + A) \cdot B]_{i,j} = \sum_{k=1}^{n} (\delta_{ik} + a_{ik}) \cdot b_{kj} &= (1 + a_{ii}) \cdot b_{ij} + \sum_{\substack{k=1 \\ k \neq i}}^{n} (0 + a_{ik}) \cdot b_{kj} \\
&= 1 \cdot b_{ij} + a_{ii} \cdot b_{ij} + \sum_{\substack{k=1 \\ k \neq i}}^{n} a_{ik} \cdot b_{kj} \\
&= b_{ij} + \sum_{k=1}^{n} a_{ik} \cdot b_{kj} = [B + A \cdot B]_{i,j.}
\end{aligned}$$

The second part is analogous. ■

It is possible to define the closure of a matrix, in a way that satisfies property 8 in the definition of closed semirings.

Definition 3.4 *Let A be a matrix of size $n \times n$, then $A^* = I_n + A + A^2 + A^3 + \ldots$, where $A^1 = A$, and $A^{k+1} = A \cdot A^k$ for $k \geq 1$. Abusing of the notation, it will be written as $A^* = I_n + \sum_{k \geq 1} A^k$.*

This definition may be intuitively explained if a matrix is seen as a relation between the elements of the semiring. In this case, the closure of the matrix is nothing but the reflexive and transitive closure of this relation, which is exactly what is stated in the definition.

A^* satisfies property 8, as stated in the next theorem.

Theorem 3.5 *Let A be a matrix of size $n \times n$. Then*

$$A^* = I_n + A \cdot A^* = I_n + A^* \cdot A.$$

Proof.

$$\begin{aligned}
I_n + A \cdot A^* &= I_n + A \cdot \left(I_n + \sum_{k \geq 1} A^k \right) \\
&= I_n + A + A \cdot \sum_{k \geq 1} A^k \qquad (1) \\
&= I_n + A + A^2 + A^3 + \ldots \\
&= A^*
\end{aligned}$$

In this proof, Lemma 3.3 is necessary to guaranty that equality 1 holds.

In an analogous way, $A^* = I_n + A^* \cdot A$. ■

Some immediate consequences of this theorem are stated in the next corollary. The proof is omitted.

Corollary 3.6 *Let A and B be matrices of size $n \times n$. Then,*

$$A \cdot A^* = A + A \cdot A^* \cdot A = A^* \cdot A$$
$$A^* = I_n + A + A \cdot A^* \cdot A$$
$$B + A \cdot A^* \cdot B = A^* \cdot B, \text{ and } B + B \cdot A^* \cdot A = B \cdot A^*. \square$$

Some closed semirings

Next, a list of closed semirings will be shown. This list is far from being exhaustive, but it should be enough for showing the characteristics and generality of this kind of algebraic structures.

When speaking of usual semirings, one of the first examples is always the *Boolean semiring* $(\{0,1\}, \vee, \wedge, 0, 1)$. If the closure operator is defined as $0^* = 1, 1^* = 1$, then the structure $B = (\{0,1\}, \vee, \wedge, ^*, 0, 1)$ is also a closed semiring, which will be also called Boolean semiring.

The closure of a matrix A defined over this semiring is equal to the transitive and reflexive closure of A. This is easily seen by the definitions.

Another interesting closed semiring is given by

$$M = (\mathbb{R}_+ \cup \{+\infty\}, Min, +, ^*, +\infty, 0),$$

where $a^* = 0$ for every $a \in \mathbb{R}_+ \cup \{+\infty\}$. The closure of a matrix defined over this semiring is the all-pair minimum cost relation between the nodes of a graph defined by the matrix. The proof of this fact may be done by induction, by showing that the matrix A^k represents the all-pair minimum cost relation associated to A, over paths of length equal to k. Then, A^* will represent the all-pair minimum cost relation on paths of any length.

The properties of the operators $Min, +$, and * make it easy to verify that this structure satisfies all the axioms of closed semirings.

The algebraic structure $K = (\mathcal{P}(\Sigma^*), \cup, \cdot, ^*, \emptyset, \{\varepsilon\})$, where Σ is an alphabet, $\cdot$ the language concatenation, ε the empty word, and * the Kleene-star, is also a closed semiring.

The closure of a matrix over this semiring is the matrix of regular expressions for the accepted languages of the automaton defined by this matrix, for all the possible ways of choosing one initial and one final state. This last statement should be clear by the end of the next section.

Note that for the three examples presented so far, the neutral element for addition is absorbing with respect to multiplication. This is the reason for which on the examples in Section 2 the solution could be obtained, without the need of having complete graphs. In all those cases, the elements in the matrices corresponding to missing edges are filled with the neutral element for addition.

Example 4.1, will present a generalization of the closed semiring M, where the neutral element for addition is not absorbing. For that closed semiring, in order to obtain the matrix that represents a graph, the weights of all edges must be given explicitly.

On the other hand, these closed semirings are constructed from usual semirings, by just adding a closure operator that fulfills property 8. Unfortunately, this is not always possible, and one must construct, for some cases, the completion of the semiring using the method described before. This is better explained with the following example.

Example 3.7 *Consider the semiring* $(\mathbb{R}, +, \cdot, 0, 1)$. *Then, there is no way to define a closure operator* *, *such that* $1^* = 1 + 1 \cdot 1^*$.

Suppose that the operator * *has been defined for* 1 *and fulfills the stated property. Then, as* 1 *is the neutral element for multiplication, it follows that*

$$1^* = 1 + 1 \cdot 1^* = 1 + 1^*,$$

and from this, $0 = 1$, *which is a contradiction because, in this particular semiring,* $0 \neq 1$.

From this follows that there is no way to define the closure operator over this semiring in such a way that fulfills property 8.□

Nevertheless, from the regular semiring $(\mathbb{R}, +, \cdot, 0, 1)$, a closed semiring may be constructed by using the completion. This process produces the structure $G = (\mathbb{R} \cup \{u\}, +, \cdot, ^*, 0, 1)$, where

$$a^* = \begin{cases} \frac{1}{1-a} & \text{if } a \notin \{1, u\} \\ u & \text{if } a \in \{1, u\}. \end{cases}$$

This last closed semiring is important, because for a matrix A defined over it, $A^* = (I - A)^{-1}$, then A^{-1} may be computed by calculating $(I - A)^*$. The validity of this calculation is based on Theorem 3.5.

This is only partially true, because the presence of the element u in the calculations may lead to incorrect results. The details on how to perform this calculations correctly will not be explained here, but can be found on [Leh77].

4 Calculating the closure

The definition of the closure of a matrix has a very intuitive meaning. Nonetheless, this definition is useless for calculating it in an automated way, for an infinite amount of matrices need to be calculated and added. This is shown in the next example.

Example 4.1 *Let* $S = (\mathbb{R} \cup \{+\infty, -\infty\}, Min, +, ^*, +\infty, 0)$ *where* $+$ *is the usual real sum with* $(+\infty) + (-\infty) = +\infty$, *and*

$$a^* = \begin{cases} 0 & \textit{if } a \geq 0 \\ -\infty & \textit{if } a < 0. \end{cases}$$

It is easy to see that S is a closed semiring.

Let now $A = \begin{pmatrix} -1 & -1 \\ -1 & -1 \end{pmatrix}$.

To calculate A^* using the definition, begin by calculating the partial sums $I_2 + \sum_{k=1}^{n} A^k$ *for every* $n \geq 1$.

Note that the operations $+$ and $\cdot$ here refer to the general semiring operators, and not to the addition and multiplication over the real numbers.

Given the following matrices

$$I_2 = \begin{pmatrix} 0 & +\infty \\ +\infty & 0 \end{pmatrix}, A, A^2 = \begin{pmatrix} -2 & -2 \\ -2 & -2 \end{pmatrix}, \ldots, A^k = \begin{pmatrix} -k & -k \\ -k & -k \end{pmatrix}, \ldots.$$

it is easy to see that for every $n \geq 1, I_2 + \sum_{k=1}^{n} A^k = \begin{pmatrix} -n & -n \\ -n & -n \end{pmatrix}$. *And thus,*

$$A^* = I_2 + \sum_{k \geq 1} A^k = \begin{pmatrix} -\infty & -\infty \\ -\infty & -\infty \end{pmatrix},$$

but this matrix is not equal to any of the partial sums. This means that it cannot be calculated in any finite number of steps by using only the definition.□

This example shows the need for an algorithm that calculates the closure of a matrix.

As it was said in Section 2, there exist algorithms for calculating the desired closure over specific closed semirings. Warshall's algorithm computes the closure of a matrix on the Boolean semiring; the Floyd-Warshall algorithm does it for matrices over the semiring denoted as M in the previous section, as Kleene's algorithm does for matrices over K. Furthermore, the Gauss-Jordan algorithm computes the closure of a matrix over G, with the advantage that it does not need the new element u given by the completion.

Then, one may try to generalize those algorithms in such a manner that it may be applied on other closed semirings. The next algorithm is a generalization of Kleene's algorithm to compute regular expressions.

Algorithm 4.2 ***Input:*** $A = [a_{ij}]_{1 \leq i,j \leq n}, a_{ij} \in S$, *$S$ a closed semiring*

1. $A_0 = A$
2. ***For*** $k = 1$ *to* n *do*
  3. ***For each*** $1 \leq i, j \leq n$ ***do***
    4. $[A_k]_{ij} \leftarrow [A_{k-1}]_{ij} + [A_{k-1}]_{ik} \cdot ([A_{k-1}]_{kk})^* \cdot [A_{k-1}]_{kj}$
5. $R = I_n + A_n$
Output: R

The proof of correctness of this algorithm is beyond the scope of this work. The interested reader will find all the details in [Leh77].

Intuitively, the algorithm may be thought as a method for traveling, in an ordered way, over the whole graph. Whenever an edge is found, the value held in the last node is multiplied to the weight associated to the edge; and at every node, the values held by all the incoming edges is added. Hence, the need of commutativity in addition.

Over this point of view, it is easy to see why, if the neutral for addition is absorbing with respect to multiplication, a non-complete graph can be thought as a complete graph where the missing edges have this neutral element as weight. When the multiplication is performed over that edge, the result will be once again the neutral element, which will not contribute to the sum performed at the next node.

5 The general solution

Returning to the line presented in the motivation, it is now time to use Algorithm 4.2 to calculate all the solutions. In fact, what will be calculated is the "most general solution" for graphs with two nodes. Every specific problem will be only an instance of this, for which a solution is already known.

The first step, as it has been pointed out several times before, is to complete the graph. Figure 4 shows the complete two-node graph.

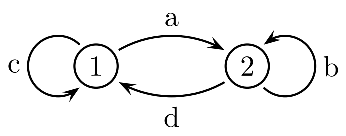

Fig. 4: A complete two-node graph

Any two-node graph will have this form; the only additional step necessary after calculating the closure of its associated matrix is to translate the results into the values corresponding to the specific problem. This is not only true for problems on graphs, but also for any problem on matrices defined over a closed semiring. In particular, it will be possible to invert a matrix, by means of the semiring G defined before.

The matrix associated to the graph in Figure 4 is

$$A = \begin{pmatrix} c & a \\ d & b \end{pmatrix} .$$

Now the algorithm must be applied to this matrix. In order to make the algorithm clear, the steps will be performed explicitly.

As a first step, the original matrix is copied into a new A_0. Now fix $k = 1$ (step 2 of the algorithm). Then, for every $1 \leq i, j \leq 2$ the value of $[A_1]_{ij}$ is given by

$$[A_0]_{ij} + [A_0]_{i1} \cdot ([A_0]_{11})^* \cdot [A_0]_{1j} = [A_0]_{ij} + [A_0]_{i1} \cdot (c)^* \cdot [A_0]_{1j} \, ,$$

as stated on algorithm's step 4.

Substituting every possible value of i and j yields

$$\begin{aligned}
[A_1]_{11} &= [A_0]_{11} + [A_0]_{11} \cdot (c)^* \cdot [A_0]_{11} \\
&= c + c \cdot (c)^* \cdot c \\
[A_1]_{12} &= [A_0]_{12} + [A_0]_{11} \cdot (c)^* \cdot [A_0]_{12} \\
&= a + c \cdot (c)^* \cdot a \\
[A_1]_{21} &= [A_0]_{21} + [A_0]_{21} \cdot (c)^* \cdot [A_0]_{11} \\
&= d + d \cdot (c)^* \cdot c \\
[A_1]_{22} &= [A_0]_{22} + [A_0]_{21} \cdot (c)^* \cdot [A_0]_{12} \\
&= b + d \cdot (c)^* \cdot a.
\end{aligned}$$

Thus, the second loop of the algorithm, with $k = 2$, will work on

$$A_1 = \begin{pmatrix} c + c \cdot (c)^* \cdot c & a + c \cdot (c)^* \cdot a \\ d + d \cdot (c)^* \cdot c & b + d \cdot (c)^* \cdot a \end{pmatrix},$$

This yields the new matrix

$$A_2 = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

where

$$\begin{aligned}
a_{11} &= c + c \cdot (c)^* \cdot c + (a + c \cdot c^* \cdot a) \cdot (b + d \cdot c^* \cdot a)^* \cdot (d + d \cdot c^* \cdot c) \\
a_{12} &= a + c \cdot (c)^* \cdot a + (a + c \cdot c^* \cdot a) \cdot (b + d \cdot c^* \cdot a)^* \cdot (b + d \cdot c^* \cdot a) \\
a_{21} &= d + d \cdot (c)^* \cdot c + (b + d \cdot c^* \cdot a) \cdot (b + d \cdot c^* \cdot a)^* \cdot (d + d \cdot c^* \cdot c) \\
a_{22} &= b + d \cdot (c)^* \cdot a + (b + d \cdot c^* \cdot a) \cdot (b + d \cdot c^* \cdot a)^* \cdot (b + d \cdot c^* \cdot a).
\end{aligned}$$

It is now possible to calculate $A^* = I_2 + A_2$, as stated in step 5.

This is the general solution for the problem. It is now possible to do all the needed translations to solve the three problems presented in the motivation, and any other that may arise later.

In order to solve the reachability problem, the Boolean semiring B will be used, as was specified before. The needed translations transform the operator $+$ into $\vee$; $\cdot$ into $\wedge$; and * into the specific operator * of B that yields 1 over any input. The edges are translated according to the following table

$$\begin{aligned}
a &= 1, \\
b &= 1, \\
c &= 0, \\
d &= 0.
\end{aligned}$$

Refer to Figure 1, to understand this translations properly.

Thus, the solution to the first problem is given by

$$\begin{aligned}
[a^*]_{11} &= 1 + c + c \cdot (c)^* \cdot c + (a + c \cdot c^* \cdot a) \cdot (b + d \cdot c^* \cdot a)^* \cdot (d + d \cdot c^* \cdot c) \\
&= 1 \vee (0 \vee 0 \wedge 1 \wedge 0) \vee (1 \vee 0 \wedge 1 \wedge 1) \wedge 1 \wedge (0 \vee 0 \wedge 1 \wedge 0) = 1 \\
[a^*]_{12} &= 0 \vee (1 \vee 0 \wedge 1 \wedge 1) \vee (1 \vee 0 \wedge 1 \wedge 1) \wedge 1 \wedge (1 \vee 0 \wedge 1 \wedge 1) = 1 \\
[a^*]_{21} &= 0 \vee (0 \vee 0 \wedge 1 \wedge 0) \vee (1 \vee 0 \wedge 1 \wedge 1) \wedge 1 \wedge (0 \vee 0 \wedge 1 \wedge 0) = 0 \\
[a^*]_{22} &= 1 \vee (1 \vee 0 \wedge 1 \wedge 1) \vee (1 \vee 0 \wedge 1 \wedge 1) \wedge 1 \wedge (1 \vee 0 \wedge 1 \wedge 1) = 1.
\end{aligned}$$

Here, the usual convention of operator $\wedge$ having a higher priority than $\vee$ is used.

Hence, the desired result is

$$A^*_B = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix},$$

which is exactly the same result shown in Section 2, as expected.

Now, the second solution is obtained using the closed semiring M defined above. The translations are required to change $+$ into Min, $\cdot$ into $+$, and the operator * into the operator that gives to any input the value 0. Furthermore, the edges are translated according to Figure 2 as follows:

$$\begin{aligned}
a &= 3, \\
b &= 1, \\
c &= \infty, \\
d &= \infty.
\end{aligned}$$

Once again, after applying all the described translations to the matrix A^*, the desired solution

$$A^*_M = \begin{pmatrix} 0 & 3 \\ \infty & 0 \end{pmatrix}$$

is obtained.

The same solution is applicable for the third example in the motivation using the closed semiring K, with the only non-trivial necessary translation being the edges c and d mapped to $\emptyset$.

The result on this closed semiring is the matrix

$$A^*_K = \begin{pmatrix} \{\varepsilon\} & \{a\} \cdot \{b\}^* \\ \emptyset & \{b\}^* \end{pmatrix}.$$

Inverting a matrix

As it has been pointed out many times before, the solution found before is the "most general", and may be translated into any specific closed semiring, in order to obtain a desired result.

Next, the closed semiring G defined in Section 3 will be used to calculate the inverse of a matrix of size 2×2.

As an example, suppose you want to invert the matrix

$$C = \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix}.$$

In this case, the application of the most general solution has to be done carefully, because the closure of a matrix C does not yield C^{-1} but $(I_2 - C)^{-1}$. Thus, the first step in order to obtain a correct result is to calculate $D = (I_2 - C)$, and only after that, will the closure be useful, as will be shown in Proposition 5.1.

Continuing with the example,

$$\begin{aligned} D &= (I_2 - C) \\ &= \begin{pmatrix} -1 & -1 \\ 0 & -1 \end{pmatrix}. \end{aligned}$$

It is now possible to compute the inverse of C, by simply finding out the matrix A^*. The correctness of this calculation is shown in the next proposition.

Proposition 5.1 *Let A be a matrix of size $n \times n$ defined over the closed semiring G be such that the element u does neither appear in A, nor in $(I_n - A)^*$. Then $(I_n - A)^* = A^{-1}$.*

Proof. Using Theorem 3.5, calculate

$$A \cdot (I_n - A)^* = A \cdot (I_n + (I_n - A) \cdot (I_n - A)^*).$$

This expression can be further extended by means of Lemma 3.3; this yields

$$\begin{aligned} A \cdot (I_n - A)^* &= A + A \cdot ((I_n - A) \cdot (I_n - A)^*) \\ &= A + A \cdot (I_n - A)^* - A \cdot A \cdot (I_n - A)^*. \end{aligned}$$

At this point, using the commutativity, and a property of this specific semiring, that is, that every element has an additive inverse, the expression above may be simplified into

$$\begin{aligned} O_n &= A - A \cdot A \cdot (I_n - A)^* \\ &= A \cdot (I_n - A \cdot (I_n - A)^*). \end{aligned}$$

This equation leads to

$$A = A \cdot A \cdot (I_n - A)^*.$$

Here the assumption that the element u does not appear in A is needed because, then, u does not appear in $B = A \cdot (I_n - A)^*$. This means that A and B are real matrices such that $A = A \cdot B$. This implies that $B = I_n$, and thus, $(I_n - A)^* = A^{-1}$. ■

It is important to note that this proof is structure-dependent. It strongly uses many of the properties of the specific closed semiring G; this means that for other closed semirings, the result may not be true.

So, the closure of the matrix can be now freely used to calculate inverses. The next table shows the result of applying it to the present example. Only the calculation of a^*_{11} is made completely, as the others are analogous.

$$
\begin{aligned}
d^*_{11} &= 1 + c + c \cdot (c)^* \cdot c + (a + c \cdot c^* \cdot a) \cdot (b + d \cdot c^* \cdot a)^* \cdot (d + d \cdot c^* \cdot c) \\
&= 1 - 1 + (-1) \cdot \frac{1}{2} \cdot (-1) + \left(-1 + (-1) \cdot \frac{1}{2} \cdot (-1)\right) \cdot \frac{1}{2} \cdot 0 \\
&= \frac{1}{2} \\
d^*_{12} &= a + c \cdot (c)^* \cdot a + (a + c \cdot c^* \cdot a) \cdot (b + d \cdot c^* \cdot a)^* \cdot (b + d \cdot c^* \cdot a) \\
&= -\frac{1}{4} \\
d^*_{21} &= d + d \cdot (c)^* \cdot c + (b + d \cdot c^* \cdot a) \cdot (b + d \cdot c^* \cdot a)^* \cdot (d + d \cdot c^* \cdot c) \\
&= 0 \\
d^*_{22} &= 1 + b + d \cdot (c)^* \cdot a + (b + d \cdot c^* \cdot a) \cdot (b + d \cdot c^* \cdot a)^* \cdot (b + d \cdot c^* \cdot a) \\
&= \frac{1}{2}.
\end{aligned}
$$

Thus, the result is

$$
C^{-1} = \begin{pmatrix} \frac{1}{2} & -\frac{1}{4} \\ 0 & \frac{1}{2} \end{pmatrix}.
$$

This can be easily verified by multiplying both matrices.

6 Conclusions

A generic algorithm for calculating the closure of matrices defined over closed semirings was presented. This algorithm may be applied in order to solve some classical problems over graphs such as reachability, or shortest-path problems.

Nonetheless, the algorithm is strong enough to solve problems that are not directly related to graphs. One example of this is calculating the inverse of matrices of real numbers.

Obviously, the range of this algorithm is not limited to the examples presented, as there are many other closed semirings for which it is desirable to calculate the matrix closure. The idea of having a general algorithm is to be able to solve different kinds of problems, without worrying about the correctness of new algorithms.

References

[Leh77] D. J. Lehmann. Algebraic Structures for Transitive Closure. *Theoretical Computer Science*, 4:59-76, 1977

[Vog04] H. Vogler. The Shortest Path Problem - Lecture Notes. Script, Dresden University of Technology, Department of Computer Science, Institute for Theoretical Computer Science, 2004.