# Transitive Closures for Relations on a Set

Michael A. Goodrich

October 24, 2020

## 1 Reflexive and Symmetric

For any relation of type "relations on $A$", we can determine whether the relation satisfies some useful properties. This section talks about the reflexive and symmetric properties of "relations on $A$; see Definitions 3–4 in section 9.1.4 of the textbook.

### 1.1 Reflexive

A relation $R$ on $A$ is reflexive if for every element $a \in A$ the tuple $(a, a) \in R$. Simply put, a relation is reflexive if every element of the underlying set $A$ is related to itself.

We have three ways to detect whether a relation is reflexive, one for each of the representations that we can use (set, matrix, graph). Consider the relation $R_3$ from Example 7 in section 9.1.4 of the textbook, rewritten to use $A = \{a, b, c, d\}$

$$R_3 = \{(a, a), (a, b), (a, d), (b, a), (b, b), (c, c), (d, a), (d, d)\}$$

#### 1.1.1 Using Sets

We can determine whether $R_3$ is reflexive for our set notation using the following pseudocode.

```
return_value = true
for each a in A
{
    t = (a,a)
    if R.find(t) == false
    {
        return_value == false
    }
}
return return_value
```

This pseudocode searches through each $a$ in the underlying set $A$, turns it into a tuple, $t = (a, a)$, and tests whether the tuple is in the set $R$ using the `R.find(t)` method. The pseudocode returns true if all elements $(a, a) \in R$ and false otherwise.

For $R_3$ above, we see that $(a, a)$, $(b, b)$, $(c, c)$, and $(d, d)$ are in $R_3$, so $R_3$ is reflexive.

### 1.1.2 Using Matrices

The matrix representation for $R_3$ is given by

$$\mathbf{M}_{R_3} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

The relation is reflexive if and only if each element on the diagonal of the matrix has a value of one. This is an easy check, both visually and in pseudocode.

### 1.1.3 Using Graphs

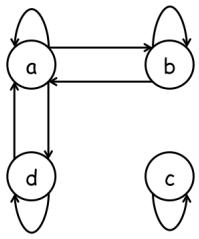


Figure 1: Graph for the $R_3$ relation.

The graph for relation $R_3$ is shown in Figure 1. $(a, a) \in R$ means that there is a self-loop on the graph. Since there is a self-loop for each vertex in the graph, we can see that $R_3$ is reflexive.

## 1.2 Symmetric

A relation $R$ on $A$ is symmetric if $\forall a, b \in A \;\; (a, b) \rightarrow (b, a)$.

### 1.2.1 Using Sets

Consider the following pseudo-code:

```
return_value = true
for each a in A
{
    for each b in A
    {
        t1 = (a,b)
        t2 = (b,a)
        if ((R.find(t1) == true) and (
```

```
        R.find(t2) == false)) or (((R.find(t1) == false) and (
        R.find(t2) == true))
        {
            return_value == false
        }
    }
}
```

Notice how the pseudocode returns false only if one of $(a, b) \in R$ or $(b, a) \in R$ but not both. If neither $(a, b)$ nor $(b, a)$ is in $R$ then there is no problem, and if both are in $R$ then there is no problem.

### 1.2.2 Using Matrices

Let's return to the matrix representation for $R_3$.

$$\mathbf{M}_{R_3} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

Symmetry can be seen by reflecting the top-right part of the matrix across the diagonal into the bottom-right. If there is a 1 in the $(i, j)^{\text{th}}$ element of the matrix, then there needs to be a 1 in the $(j, i)^{\text{th}}$ element.

For those of you who have taken or are taking linear algebra, we can test this using only matrix operators using by "XOR-ing" the matrix and its transpose,

$$\mathbf{M}_{R_3} \oplus \mathbf{M}_{R_3}^T.$$

If there are any 1's in the resulting matrix, then the relation is not symmetric.

### 1.2.3 Using Graphs

I think the matrix approach is pretty easy, but I've used matrices a lot. Many of you have not, so it probably looks yucky. Fortunately, the graph approach is really easy.

A relation is symmetric if and only if any directed edge between vertex $a$ and vertex $b$ has a corresponding directed edge between vertex $b$ and vertex $a$. From the graph in Figure 1, there are edges in both directions for $a, b$ and $a, d$, so the relation is symmetric.

## 2 Reflexive and Symmetric Closures

Definition 1 in section 9.4.2 of the textbook gives the definition of a closure. The idea is to state some property that you'd like your relation to have and then **create a new relation from the old relation by *adding new things* but *without adding too much***.

Let $R$ be the relation, and closure$(R; P)$ be an awkward notation for the relation that is created when we close $R$ so that it satisfies property $P$. Consider my interpretation of the definition of a closure, given in the boldface type in the previous paragraph. You can see the "adding new things" part of the definition of a closure in the statement in the statement "[the new relation] contains $R$," which means $R \subseteq$ closure$(R; P)$.

Consider again my interpretation of the definition of a closure, given in the boldface type from two paragraphs ago. The "without adding too much" of interpretation is stated in Definition 1 as "[the new relation] is a subset of every subset of $A \times A$ containing $R$ with property **P**." This is really tough to parse, so let's talk about the Example 1 and Example 2 from section 9.4.2 to see if we can make sense of it.

The relation that they are considering is

$$R = \{(a, a), (a, b), (b, a), (c, b)\}$$

where $A = \{a, b, c\}$. The matrix representation is given by

$$\mathbf{M}_R = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Let's do a quick "sanity check". There are four tuples in the set representation for $R$ and four ones in the matrix representation; that suggests that we got the matrix representation correct.

The graph representation for $R$ is given in Figure 2.

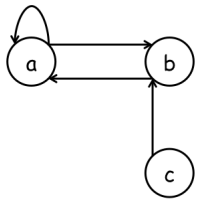


Figure 2: Graph for the $R$ relation.

## 2.1 Reflexive Closure

There are a lot of reflexive relations that contain $R$. Here are a few:

$$\mathbf{M}_{R'} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

and

$$\mathbf{M}_{R''} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

and

$$\mathbf{M}_{R'''} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

We can check to see that each of these relations are reflexive by seeing that there are ones in every location on the diagonal. Let $R^{'}$ be the corresponding relation derived from $\mathbf{M}_{R'}$, and likewise for $R^{''}$ and $R^{'''}$ from $\mathbf{M}_{R''}$ and $\mathbf{M}_{R'''}$, respectively.

Of the three closures, only $R^{'''}$ is the correct reflexive closure. This closure only added $(b, b)$ and $(c, c)$ to the relation, and nothing more. By contrast, $R^{'}$ added ones everywhere, and $R^{''}$ added an extra one on row 2, column 3. Notice how $R \subset R'$, $R \subset R^{''}$, and $R \subset R^{'''}$; each of these relations adds enough new things to make the new relation reflexive (they satisfy the reflexive property). But $R^{'''} \subset R^{''}$ and $R^{'''} \subset R^{'}$, which means that $R^{'''}$ is contained in the other relations. This means that the other relations added too much. Indeed, there is no smaller relation that contains $R$ and is still reflexive.

The graph of the reflexive closure is shown in Figure 3. In keeping with the interpretation that we should

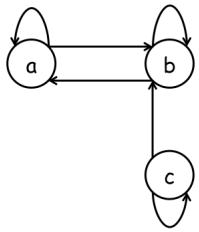


Figure 3: Graph for the reflexive closure of the $R$ relation.

"add enough new things to satisfy the property, but not too much", we could have added a lot of edges to the graph, but we added the fewest edges that yielded the reflexive properties – the two self-loops on $b$ and $c$. Many other graphs could also have been used, but they would have had unneeded edges added. The reasoning is the same as for the matrix representation.

## 2.2 Symmetric Closure

Recall that we are considering the relation

$$R = \{(a, a), (a, b), (b, a), (c, b)\}$$

with the matrix representation given by

$$\mathbf{M}_R = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

and the graph representation given by Figure 2. The idea of a symmetric closure is that we want to add enough tuples to make sure that for every $(a, b) \in R$ then $(b, a)$ is also a tuple in $R$. We want to add enough tuples, but the minimum number possible to satisfy the symmetry property.

The symmetric closure is

$$\texttt{closure}(R; \texttt{symmetry}) = \{(a, a), (a, b), (b, a), (c, b), (b, c)\}$$

in which we added the tuple $(b, c)$ since $(c, b)$ was already in the relation.

If we use the matrix approach, we get

$$\mathbf{M}_{\texttt{closure}(R;\texttt{symmetry})} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & \mathbf{1} \\ 0 & 1 & 0 \end{bmatrix}$$

where the bold-faced blue 1 was added to ensure symmetry.

If we use the graph approach, we produce Figure 4 in which the red edge was added.

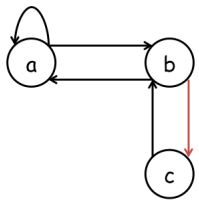


Figure 4: Graph for the symmetric closure of the $R$ relation.

# 3 Transitive Closures

A relation is transitive if for all $(a, b) \in R$ and $(b, c) \in R$ it follows that $(a, c) \in R$. Notice that this is an implication.

Producing the transitive closure of a relation is much more challenging than producing the reflexive or symmetric closures. The textbook spends a great deal of time deriving an algorithm that produces the transitive closure. We'll discuss three algorithms: a set-based algorithm, a matrix-based algorithm, and a graph-based algorithm.

We'll use the following relation as our example:

$$R = \{(a, a), (a, b), (b, a), (b, b), (c, d), (d, a), (d, d)\},$$

which is a relation on the set on set $A = \{a, b, c, d\}$.

The matrix representation for this relation is give by

$$\mathbf{M}_R = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

The graph for this relation is shown in Figure 5

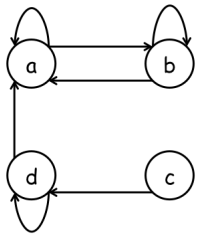


Figure 5: Graph for the relation used to illustrate transitive closure.

### 3.1 Set-Based Algorithm

The set-based algorithm for computing the transitive closure is:

```
Compute Transitive Closure of R:
   R'=R
   Repeat until no change:
       for each (x,y) and (y,z) in R', add (x,z) to R'
```

Look carefully at what is common in the three tuples. We start with $(x, \mathbf{y})$ and $(\mathbf{y}, z)$. These two tuples have the element $\mathbf{y}$ in common, the once as the first element of a tuple and then again as the second element of the other tuple. Transitivity then takes the remaining elements of the tuple and puts them together to create $(x, z)$.

The reason that this process is repeated is because adding $(x, z)$ to $R$ may then create another tuple pair that needs to be made transitive. Let's work the example.

For the relation

$$R = \{(a, a), (a, b), (b, a), (b, b), (c, d), (d, a), (d, d)\},$$

Please make a mental note that there are $|R| = 7$ tuples in the relation.

we'll find all pairs of tuples that have an element in common, with the element in the first location in one of the tuples and in the second location in the other. Here are the pairs for this relation:

$$\{(a, a), (a, b)\}$$
$$\{(a, b), (b, a)\}$$
$$\{(a, b), (b, b)\}$$
$$\{(b, a), (a, a)\}$$
$$\{(b, a), (a, b)\}$$
$$\{(b, b), (b, a)\}$$
$$\{(c, d), (d, a)\}$$
$$\{(c, d), (d, d)\}$$
$$\{(d, a), (a, a)\}$$
$$\{(d, a), (a, b)\}$$
$$\{(d, d), (d, a)\}$$

The way that I listed the tuples in the set emphasizes the pattern. The second element in the first tuple in the set always matches the first element of the second tuple in the set. One by one, we'll need to add a tuple to the relation.

$$\begin{aligned}
\{(a,a),(a,b)\} &\rightarrow \texttt{R.add}((a,b)) \\
\{(a,b),(b,a)\} &\rightarrow \texttt{R.add}((a,a)) \\
\{(a,b),(b,b)\} &\rightarrow \texttt{R.add}((a,b)) \\
\{(b,a),(a,a)\} &\rightarrow \texttt{R.add}((b,a)) \\
\{(b,a),(a,b)\} &\rightarrow \texttt{R.add}((b,b)) \\
\{(b,b),(b,a)\} &\rightarrow \texttt{R.add}((b,a)) \\
\{(c,d),(d,a)\} &\rightarrow \texttt{R.add}((c,a))* \\
\{(c,d),(d,d)\} &\rightarrow \texttt{R.add}((c,d)) \\
\{(d,a),(a,a)\} &\rightarrow \texttt{R.add}((d,a)) \\
\{(d,a),(a,b)\} &\rightarrow \texttt{R.add}((d,b))* \\
\{(d,d),(d,a)\} &\rightarrow \texttt{R.add}((d,a))
\end{aligned}$$

where I used `R.add()` to indicate that I'm adding those tuples to the relation. Since sets don't have repeat elements, the uniquely added elements are stared above.

My relation after one iteration of adding things looks like

$$R_{1-\text{step}} = \{(a,a),(a,b),(b,a),(b,b),(c,d),(d,a),(d,d),(c,a)*,(d,b)*\},$$

where again I've starred the new relations. Please note that there are now $|R_{1-\text{step}}| = 9$ tuples in the relation.

It is possible that by adding $(c,a)$ and $(d,b)$ to the relation that I've created a need to add some more tuples to ensure transitivity. I'm not going to repeat the work that I've already done, but instead only consider pairs of tuples that involve one of the new pairs. This yields,

$$\begin{aligned}
\{(c,a),(a,a)\} &\rightarrow \texttt{R.add}((c,a)) \\
\{(c,a),(a,b)\} &\rightarrow \texttt{R.add}((c,b))* \\
\{(d,b),(b,a)\} &\rightarrow \texttt{R.add}((d,a)) \\
\{(d,b),(b,b)\} &\rightarrow \texttt{R.add}((d,b)) \\
\{(c,d),(d,b)\} &\rightarrow \texttt{R.add}((c,b))* \\
\{(d,d),(d,b)\} &\rightarrow \texttt{R.add}((d,b))
\end{aligned}$$

My relation now looks like:

$$R_{2-\text{steps}} = \{(a,a),(a,b),(b,a),(b,b),(c,d),(d,a),(d,d),(c,a),(d,b),(c,b)*\},$$

which has $|R_{2-\text{steps}}| = 10$ tuples.

Since there are more tuples after two steps than there were after one step, we have to continue. Again, saving space, I'll only consider pairs that involve the newly added tuple. This gives:

$$\begin{aligned}
\{(c,b),(b,a)\} &\rightarrow \texttt{R.add}((c,a)) \\
\{(c,b),(b,b)\} &\rightarrow \texttt{R.add}((c,b))
\end{aligned}$$

No new tuples were added to the relation,

$$R_{3-\text{steps}} = \{(a,a),(a,b),(b,a),(b,b),(c,d),(d,a),(d,d),(c,a),(d,b),(c,b)\},$$

and $|R_{2-\text{steps}}| = |R_{3-\text{steps}}| = 10$. We say that we have reached a *fixed point*, and that is how we tell it is time to stop the algorithm.

## 3.2 Graph-Based Algorithm

Let's look at the graph from $R_{3-\text{steps}}$, shown in Figure 6. What is different between Figure 5 (the graph

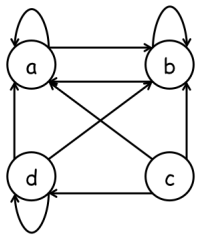


Figure 6: Graph for the relation after the transitive closure.

before the closure) and Figure 6 (the graph after the transitive closure)? The key to seeing the differences is to think about paths through the graph.

A path through a graph from one vertex to another is the sequence of directed edges that are traversed along the way. The idea of the transitive closure is that, if there is a path from some vertex $v_1$ and another vertex $v_3$ that travels through vertex $v_2$, then we should add an edge directly between vertex $v_1$ and $v_3$. Since edges are described as tuples of vertices, this is the same thing as saying if there is an edge $(v_1, v_2)$ and another edge $(v_2, v_3)$ then the transitivity property says that we should have an edge $(v_1, v_3)$. In graph terms, the transitivity property is the same as finding a path between vertices via another vertex.

Thus, the graph method for finding the transitive closure iteratively adds edges between pairs of vertices that can be reached via another vertex. Let's do this.

Figure 7 shows the steps. Begin in the upper left with the original graph. Notice that there is a path from $d$ to $b$ via $a$. Thus, we need to add a directed vertex directly from $d$ to $b$. Notice also that there is a path from $c$ to $a$ through $d$. Thus, we add a directed vertex directly from $c$ to $a$. These are all the paths from one vertex to another via a third vertex. The resulting graph is in the top right.

But now we have a new path, one between $c$ and $b$ via $a$. We add the edge $(c, b)$. There are no more edges to add. The resulting graph is in the bottom left of the figure.

We can try again, but no new edges will be added. We've reached a fixed point.

Notice that the edges that are added going from the top left to the top right of the graph are precisely those new tuples added in the first step of the set-based algorithm. Notice that the edge added going from the top right to the bottom left is the edge that is added in the second step of the set-based algorithm.

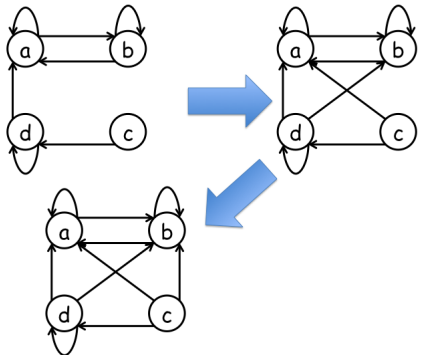


Figure 7: Iteratively adding edges to graphs for the relation to produce the transitive closure.

Cumulatively, we've found all paths between any two nodes via any other nodes. Notice, for example, that there is not an edge from $b$ to $c$ in the transitive closure precisely because there is no path via any other edge that allows you to move from $b$ to $c$.

## 3.3 Matrix-Based Algorithm

My favorite algorithm is the matrix-based algorithm. It's a bit more complex to understand because it requires some understanding of matrices, but it's so easy to write mathematically. The algorithm is in Theorem 3, section 9.4.4 of the textbook. You can ignore the algorithm. I'll only test you or give you homework on the set-based algorithm, but the graph-based algorithm will probably help you build intuition.

For those still reading, I actually don't do the algorithm in Theorem 3 of the textbook. I replace every zero-one operation (defined in Section 2.6.4 of the textbook) with regular matrix multiplication. Thus, $\mathbf{M}_R^{[i]}$ just becomes $\mathbf{M}_R * \mathbf{M}_R * \cdots * \mathbf{M}_R$, $i$ times. Then, instead of using the OR operator, I just add the matrices. When I'm done, I replace every non-zero number with a one and get the same answer.