Relations, Matrices, and Relational Composition

Part 1: Relations as Mathematical Objects

Before we can understand why matrix multiplication computes relational composition, we need to be clear about what a relation is and how to represent it.

What Is a Relation?

A *relation* is a set of ordered pairs. Nothing more, nothing less. When we say that person A advises person B, we are asserting that the pair (A, B) belongs to the advising relation. The relation itself is simply the collection of all such pairs.

Consider four people in a small organization: Ann, Bob, Cal, and Dee. Suppose the advising relationships are as follows: Ann advises Bob, Ann advises Cal, Bob advises Dee, and Cal advises Dee. The advising relation is the set:

$$\text{Advises} = \{ (\text{Ann}, \text{Bob}), (\text{Ann}, \text{Cal}), (\text{Bob}, \text{Dee}), (\text{Cal}, \text{Dee}) \}$$

Each ordered pair (x, y) in this set represents one instance of x advising y. The order matters: (Ann, Bob) means Ann advises Bob, not the reverse.

The Relational Inverse

Every relation has an *inverse*, which reverses the direction of the relationship. If the relation R means "is the boss of," then the inverse R′ means "is a subordinate of" (or equivalently, "is bossed by"). If x is the boss of y, then y is a subordinate of x.

The key logical point: if xRy is true, then yR′x is true. The pair gets reversed. Whatever x does to y in the original relation, y does to x in the inverse.

For our Advises relation, the inverse is "is advised by" or "receives advice from." Since Ann advises Bob, it follows that Bob is advised by Ann. Since (Ann, Bob) is in the Advises relation, (Bob, Ann) is in the Advised-by relation.

Forming the inverse means reversing every ordered pair:

$$\text{Advised-by} = \{ (\text{Bob}, \text{Ann}), (\text{Cal}, \text{Ann}), (\text{Dee}, \text{Bob}), (\text{Dee}, \text{Cal}) \}$$

The inverse is not independent information. It is the same facts expressed from the opposite perspective. If we know who advises whom, we automatically know who is advised by whom.

Part 2: Relational Composition

Relations can be combined to form new relations. The most important operation is composition.

The Core Idea

Suppose we have two relations, R and S. The *composition* of R and S, written R◦S (or sometimes RS), is a new relation defined as follows:

> (x, z) is in R◦S if and only if there exists some y such that (x, y) is in R and (y, z) is in S.

In plain language: x is related to z through the composed relation if there is an intermediary y such that x is R-related to y and y is S-related to z.

A Concrete Example

Let us add a second relation to our organization: Manages. Suppose Ann manages Bob, and Bob manages Cal and Dee. As a set of pairs:

$$\text{Manages} = \{ (\text{Ann}, \text{Bob}), (\text{Bob}, \text{Cal}), (\text{Bob}, \text{Dee}) \}$$

Now consider the composition Advises ◦ Manages. A pair (x, z) belongs to this composed relation if there exists some y such that x advises y and y manages z.

Let us check each possibility systematically. We need to find cases where someone advises a person who manages someone else.

> Ann advises Bob. Bob manages Cal and Dee. So (Ann, Cal) and (Ann, Dee) are in the composition.
>
> Ann advises Cal. Cal manages nobody. No contribution.
>
> Bob advises Dee. Dee manages nobody. No contribution.
>
> Cal advises Dee. Dee manages nobody. No contribution.

Therefore:

$$\text{Advises} \circ \text{Manages} = \{ (\text{Ann}, \text{Cal}), (\text{Ann}, \text{Dee}) \}$$

This composed relation might be called "advises the manager of" or "advises someone who manages." Ann advises the manager of both Cal and Dee (that manager being Bob).

The Intermediary Is the Key

The essential logic of relational composition is the search for intermediaries. To determine whether (x, z) belongs to R◦S, we must check every possible intermediary y. If even one y satisfies both conditions (x R y and y S z), then x and z are related through the composition.

This will become important shortly. The reason matrix multiplication computes relational composition is precisely that matrix multiplication performs this search over all possible intermediaries.

Renaming or Chunking Compound Relations

Once we have defined a composed relation, we can give it a name and use it as a building block for further compositions. This chunking simplifies complex expressions and connects abstract compositions to familiar concepts.

Kinship provides a rich set of examples. Let P denote the "is parent of" relation, so xPy means x is a parent of y. The relational inverse P' is "is child of": xP'y means x is a child of y. From just P and P', we can construct the entire kinship system.

Start with grandparent: if x is a parent of someone who is a parent of y, then x is a grandparent of y. In relational notation, Grandparent = PP. Conversely, Grandchild = P'P'.

Now consider the sibling relation. Two people are siblings if they share a parent. Person x is a sibling of y if x is a child of someone who is a parent of y. That is, if S stands for sibling, then S = P'P. (Strictly speaking, this includes being a "sibling" of oneself, since everyone is a child of their own parent. We set this aside for now.)

Once we have named the sibling relation, we can use it to build further. An aunt or uncle is a sibling of a parent: Uncle/Aunt = SP, which expands to P'PP. A niece or nephew is a child of a sibling: Niece/Nephew = P'S = P'P'P.

A first cousin is a child of a parent's sibling: If C is for cousin, then C = P'SP = P'P'PP. Reading left to right: from x, go up two generations to a grandparent, then down a different line to reach y." That child x is y's first cousin. Knowing that PP is grandparent, which we can write as a G, we can simplify to C = G’G. In other words xG’Gy means that x is a grandchild of someone who is the grandparent of x. People with the same grandparents are cousins (or siblings, of course).

Notice that every kinship term we have defined uses only P and P'. The apparent complexity of kinship dissolves into chains of parent and child links. The named relations (Sibling, Grandparent, Cousin) are convenient shorthand—chunks that let us reason at a higher level without writing out the full composition each time.

Part 3: Representing Relations as Matrices

A relation on a finite set can be represented as a matrix. If our set has n elements, we create an n × n matrix where the entry in row i, column j indicates whether (i, j) belongs to the relation.

The Adjacency Matrix

Using our four people (Ann, Bob, Cal, Dee) numbered 1 through 4, we can write the Advises relation as:

Advises	**Ann**	**Bob**	**Cal**	**Dee**
Ann	0	1	1	0
Bob	0	0	0	1

Cal	0	0	0	1
Dee	0	0	0	0

The 1 in row Ann, column Bob indicates that (Ann, Bob) is in the relation (Ann advises Bob). The 0 in row Ann, column Ann indicates that Ann does not advise herself.

Similarly, the Manages relation:

Manages	**Ann**	**Bob**	**Cal**	**Dee**
Ann	0	1	0	0
Bob	0	0	1	1
Cal	0	0	0	0
Dee	0	0	0	0

Transposition Is the Matrix Equivalent of Relational Inversion

If we transpose a matrix (swap rows and columns), we get the matrix for the inverse relation. This follows directly from the logic of inversion.

Recall: if xRy, then yR′x. In matrix terms: if the original matrix has a 1 in row x, column y (indicating xRy), then the inverse relation has the pair (y, x), which means the inverse matrix needs a 1 in row y, column x. Transposition does exactly this: it moves every entry from position (x, y) to position (y, x).

So if A is the adjacency matrix for a relation R, then A^T (the transpose of A) is the adjacency matrix for R′ (the inverse of R).

Part 4: Matrix Multiplication

We now turn to the mechanics of matrix multiplication, building up from the simplest case.

Multiplying a Matrix by a Vector

Suppose we have a matrix A and a column vector v. To multiply them, we compute a new vector where each entry is the result of combining one row of the matrix with the vector.

Consider this small example. We have a 3×3 matrix A and a 3×1 vector v:

$$A = [2\ 1\ 0;\ 0\ 3\ 1;\ 1\ 0\ 2] \qquad v = [1;\ 2;\ 3]$$

To compute the first entry of the product Av, we take the first row of A, which is [2, 1, 0], and combine it with v by multiplying corresponding elements and summing:

$$(2 \times 1) + (1 \times 2) + (0 \times 3) = 2 + 2 + 0 = 4$$

For the second entry, we use the second row [0, 3, 1]:

$$(0 \times 1) + (3 \times 2) + (1 \times 3) = 0 + 6 + 3 = 9$$

And for the third entry, using the third row [1, 0, 2]:

$$(1 \times 1) + (0 \times 2) + (2 \times 3) = 1 + 0 + 6 = 7$$

So Av = [4; 9; 7].

This operation is called a sum of products or dot product: multiply corresponding elements, then add.

Multiplying Two Matrices

Multiplying two matrices is simply repeating the above procedure multiple times. If B is a matrix rather than a single vector, each column of B is treated as a separate vector. The result is a matrix where each column is the product of A with the corresponding column of B.

Equivalently, we can think of it this way: to find the entry in row i, column j of the product AB, we take row i of A, take column j of B, and compute their sum of products.

This means: $(AB)ij = \sum k\ Aik \times Bkj$

We sum over all values of k. For each k, we multiply the entry from row i of A by the entry from column j of B, then add up all these products.

Part 5: Why Matrix Multiplication Computes Relational Composition

We are now ready for the central insight. We will show that if A and B are adjacency matrices representing relations, then the matrix product AB represents the composition of those relations.

The Claim

Let A be the adjacency matrix for relation R, and let B be the adjacency matrix for relation S. Then the product AB is the adjacency matrix for the composed relation $R \circ S$ (though we may need to convert positive values to 1 to get a binary matrix).

The Proof (By Unpacking the Definitions)

Recall the definition of relational composition: (i, j) belongs to $R \circ S$ if and only if there exists some k such that (i, k) belongs to R and (k, j) belongs to S.

Now recall how we compute entry (i, j) of the matrix product AB:

$$(AB)ij = \sum k\ Aik \times Bkj$$

Consider what each term $Aik \times Bkj$ represents:

Aik is 1 if (i, k) is in relation R, and 0 otherwise.

Bkj is 1 if (k, j) is in relation S, and 0 otherwise.

Their product Aik × Bkj is 1 only if both are 1.

In other words, Aik × Bkj = 1 precisely when k is an intermediary linking i to j through the two relations: i is R-related to k, and k is S-related to j.

The sum ∑k Aik × Bkj adds up the contributions from all possible intermediaries. If no such intermediary exists, every term is 0, so the sum is 0. If at least one intermediary exists, the sum is positive.

This is exactly the definition of relational composition. The matrix multiplication formula (sum over all k) is literally checking every possible intermediary. The (i, j) entry of the product is positive if and only if there exists some k connecting i to j through both relations.

A Worked Example

Let us verify this with our Advises and Manages matrices. We will compute Advises × Manages and see if it matches the composition we calculated earlier.

We already determined that Advises ◦ Manages = { (Ann, Cal), (Ann, Dee) }. Let us see if matrix multiplication agrees.

Consider the (Ann, Cal) entry. We need:

$$\sum_k \text{Advises}(\text{Ann},k) \times \text{Manages}(k,\text{Cal})$$

Going through each possible intermediary k:

k = Ann: Advises(Ann,Ann)=0, so this term is 0.

k = Bob: Advises(Ann,Bob)=1 and Manages(Bob,Cal)=1, so this term is 1.

k = Cal: Advises(Ann,Cal)=1 but Manages(Cal,Cal)=0, so this term is 0.

k = Dee: Advises(Ann,Dee)=0, so this term is 0.

Sum = 0 + 1 + 0 + 0 = 1. The entry is positive, confirming (Ann, Cal) is in the composition. Moreover, it tells us there is exactly one intermediary (Bob) linking Ann to Cal.

Now consider (Bob, Cal):

k = Ann: Advises(Bob,Ann)=0, term is 0.

k = Bob: Advises(Bob,Bob)=0, term is 0.

k = Cal: Advises(Bob,Cal)=0, term is 0.

k = Dee: Advises(Bob,Dee)=1 but Manages(Dee,Cal)=0, term is 0.

Sum = 0. So (Bob, Cal) is not in the composition, which matches our earlier finding.

The Key Insight Restated

The reason matrix multiplication computes relational composition is not a coincidence or a convenient trick. It is a direct consequence of how both operations are defined:

> Relational composition asks: is there an intermediary connecting i to j?
>
> Matrix multiplication checks every possible intermediary by summing over k.

The sum-of-products structure of matrix multiplication ($\sum_k A_{ik} \times B_{kj}$) is mathematically identical to the existential check in relational composition ($\exists$ k such that iRk and kSj). For binary matrices, the product $A_{ik} \times B_{kj}$ acts as a logical AND, and the sum over k acts as a logical OR (or a count of how many intermediaries exist).

Part 6: Counting Versus Existence

One detail deserves clarification. Matrix multiplication does not merely tell us whether an intermediary exists. It counts how many intermediaries exist.

If the (i, j) entry of AB equals 3, it means there are three different nodes k such that i is R-related to k and k is S-related to j. This is often useful information. But if we only care about whether any path exists (the pure relational composition question), we can dichotomize the result, converting all positive values to 1.

When we square an adjacency matrix (multiply it by itself), entry (i, j) tells us the number of walks of length 2 from i to j. Each intermediate node that connects them adds 1 to the count.

Summary

A relation is a set of ordered pairs. Relational composition creates a new relation by linking pairs through an intermediary. An adjacency matrix represents a relation by placing 1 in cell (i, j) when (i, j) is in the relation.

Matrix multiplication computes relational composition because the formula $(AB)_{ij} = \sum_k A_{ik} \times B_{kj}$ explicitly checks every possible intermediary k. Each term asks: does k link i to j through both relations? The sum aggregates these checks into a count.

This is not a shortcut or approximation. Matrix multiplication and relational composition are the same operation expressed in different notations. Understanding this equivalence is foundational for network analysis, where we frequently compose relations to uncover indirect connections, shared memberships, and multi-step pathways.