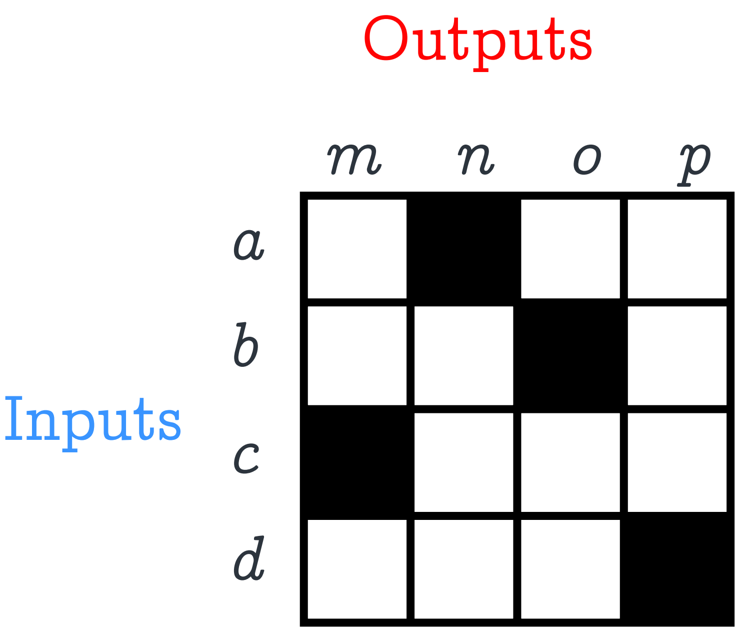
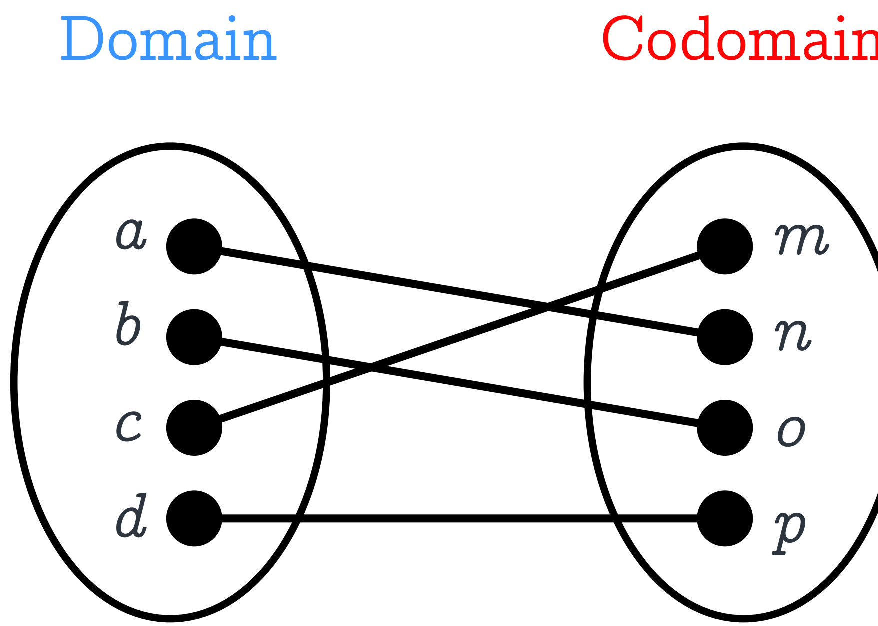
Visualizing

Matrices as Functions

v0.01



Matrix



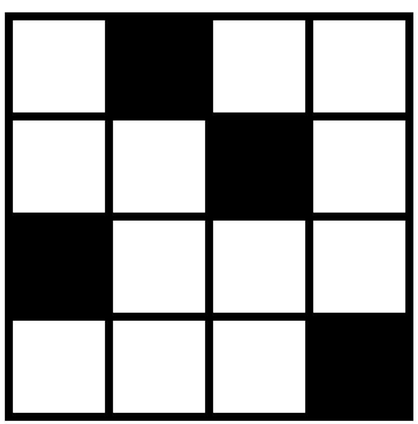
Bipartite Graph

@prabros

Matrices as Pixel Grids

A matrix is a grid of numbers arranged in rows and columns. We will consider a matrix composed entirely of 0s and 1s called the boolean matrix.

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

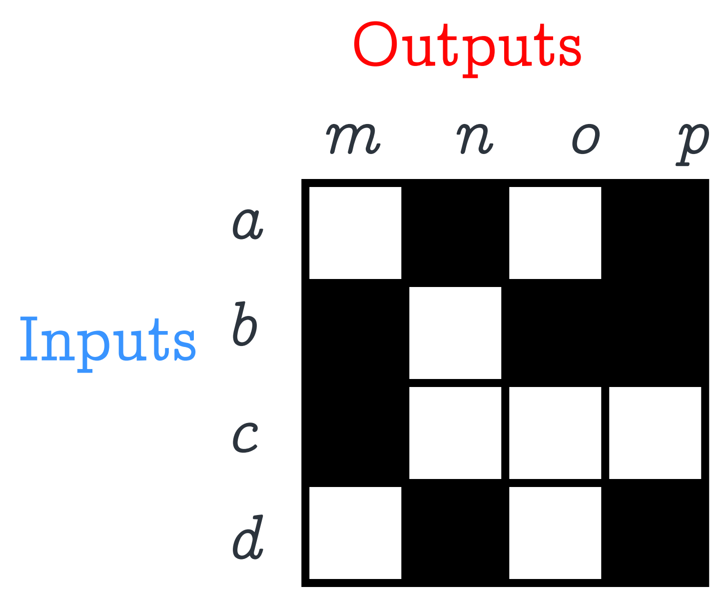


Let us use light and dark colors to represent 0s and 1s to get a pixel grid representation of the matrix.

Matrix Mappings

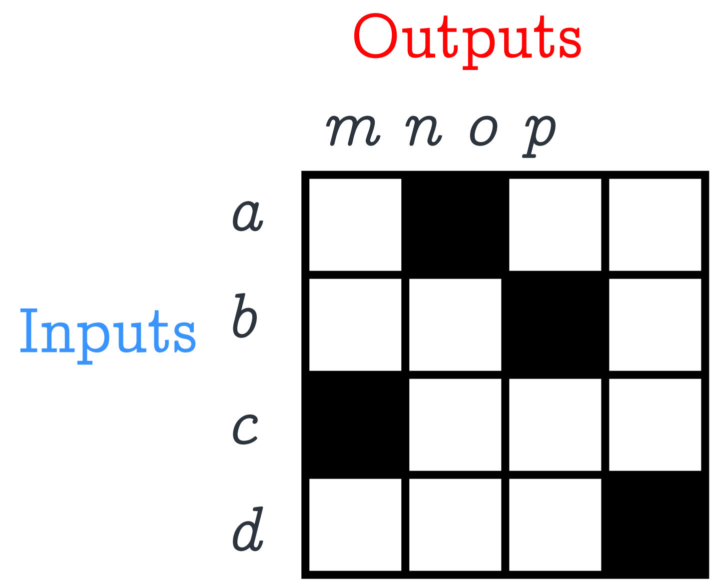
With labels on the rows and columns we can understand a matrix as representing a relational mapping. Row indices $\{\ a,\ b,\ c,\ d\ \}$ represent inputs and column indices $\{\ m,\ n,\ o,\ p\ \}$ represent outputs.

A dark square means that a mapping from input to output is present and the light square means no mapping. For example the dark square on the row index a and column index n indicates that there exists a mapping from input a to output n.



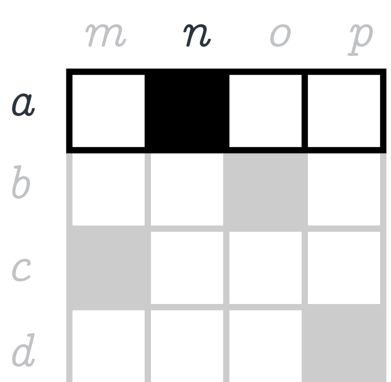
Functional Matrices

In general, a boolean matrix is equivalent to a relation. But since we are attempting to study functions in this series, we restrict our case to a matrix where there will only be a single dark square per row. This puts the restriction that there is only a single output for an input.

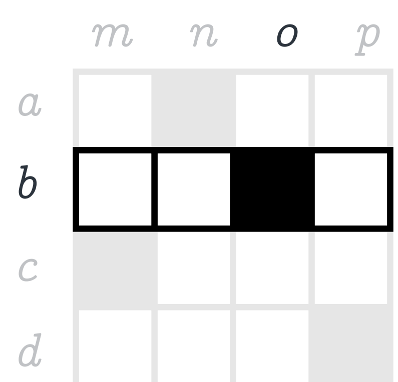


Rows vs. Columns

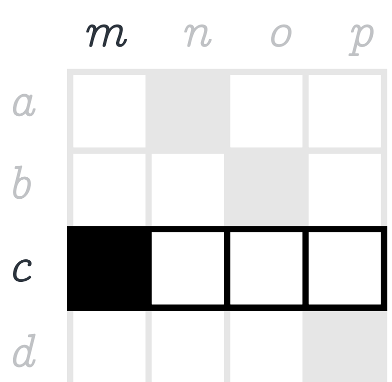
Considered this way we can understand this matrix as composed of the following mappings:



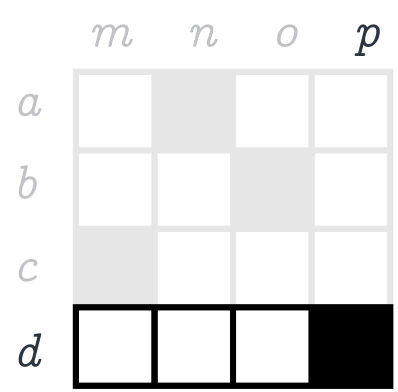
$a \mapsto n$



$b \mapsto o$



$c \mapsto m$



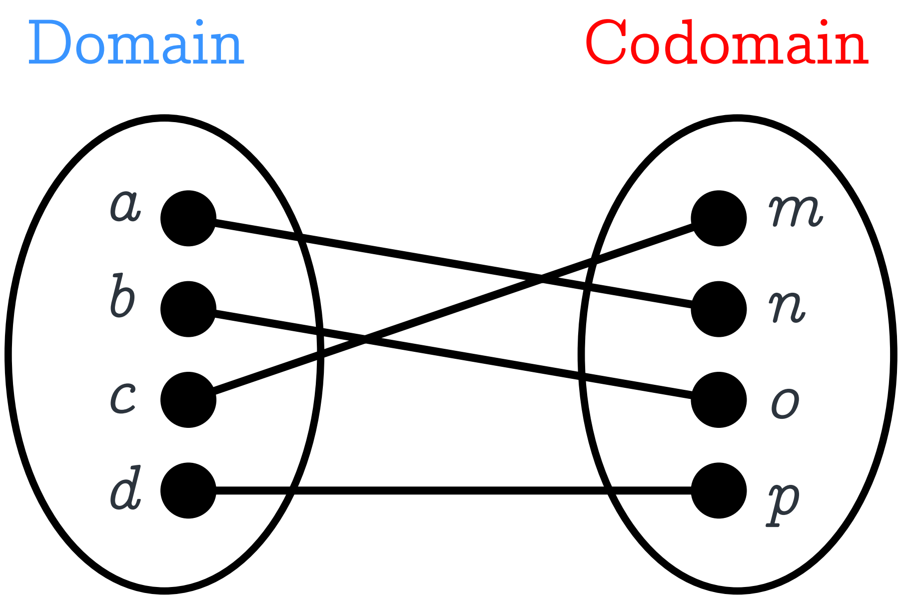
$d \mapsto p$

Set of all mappings: $\{\ a \mapsto n\ ,\ b \mapsto o\ ,\ c \mapsto m\ ,\ d \mapsto p\ \}$

Interpreting Matrix as a Bipartite Graph

This relationship between inputs and outputs can be used to give an interpretation of the matrix as a bipartite graph.

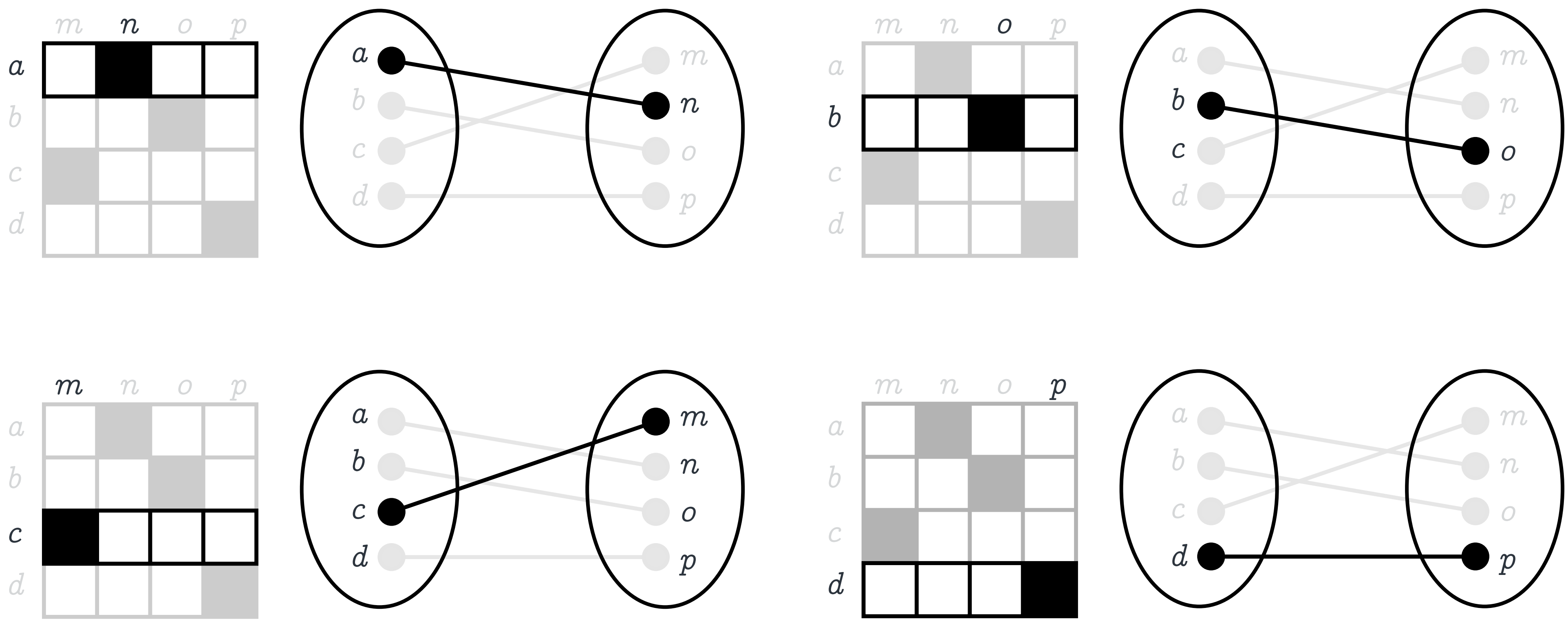
$$\{\ a \mapsto n\ ,\ b \mapsto o\ ,\ c \mapsto m\ ,\ d \mapsto p\ \}$$



A mapping say from a to n can be denoted as a line going from an element a in the domain to an element n in the codomain.

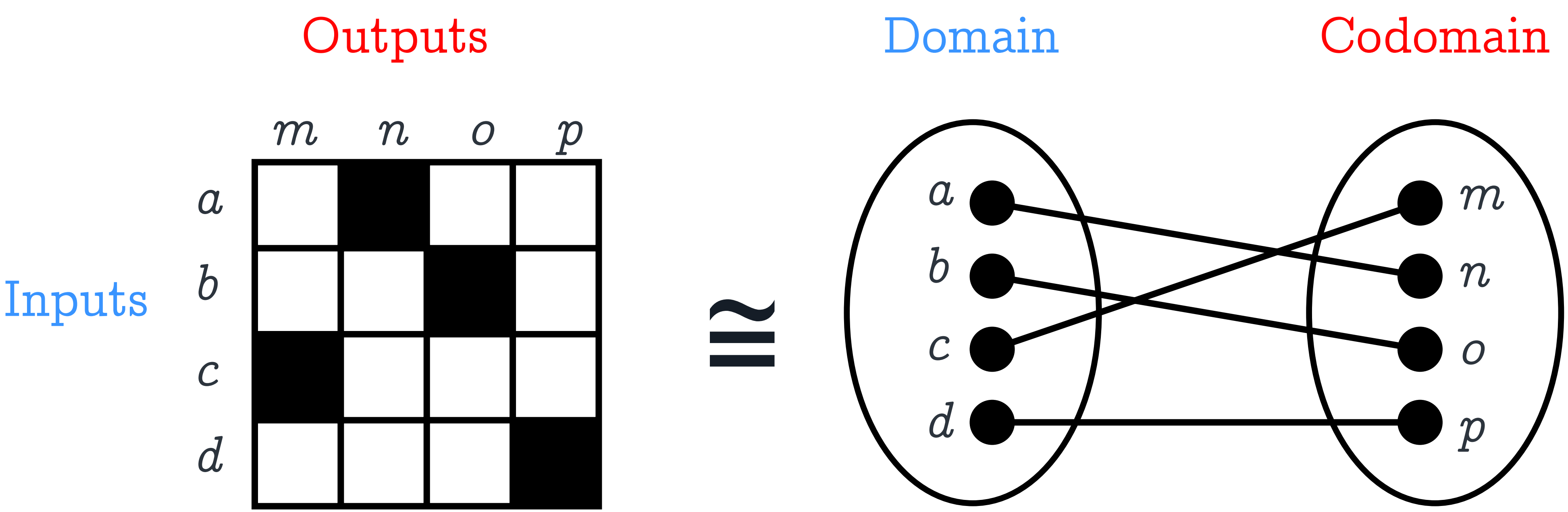
Walkthrough of the Mappings

We can walkthrough each of the rows and see how the corresponding mappings can be recognized in the bipartite graph representation.



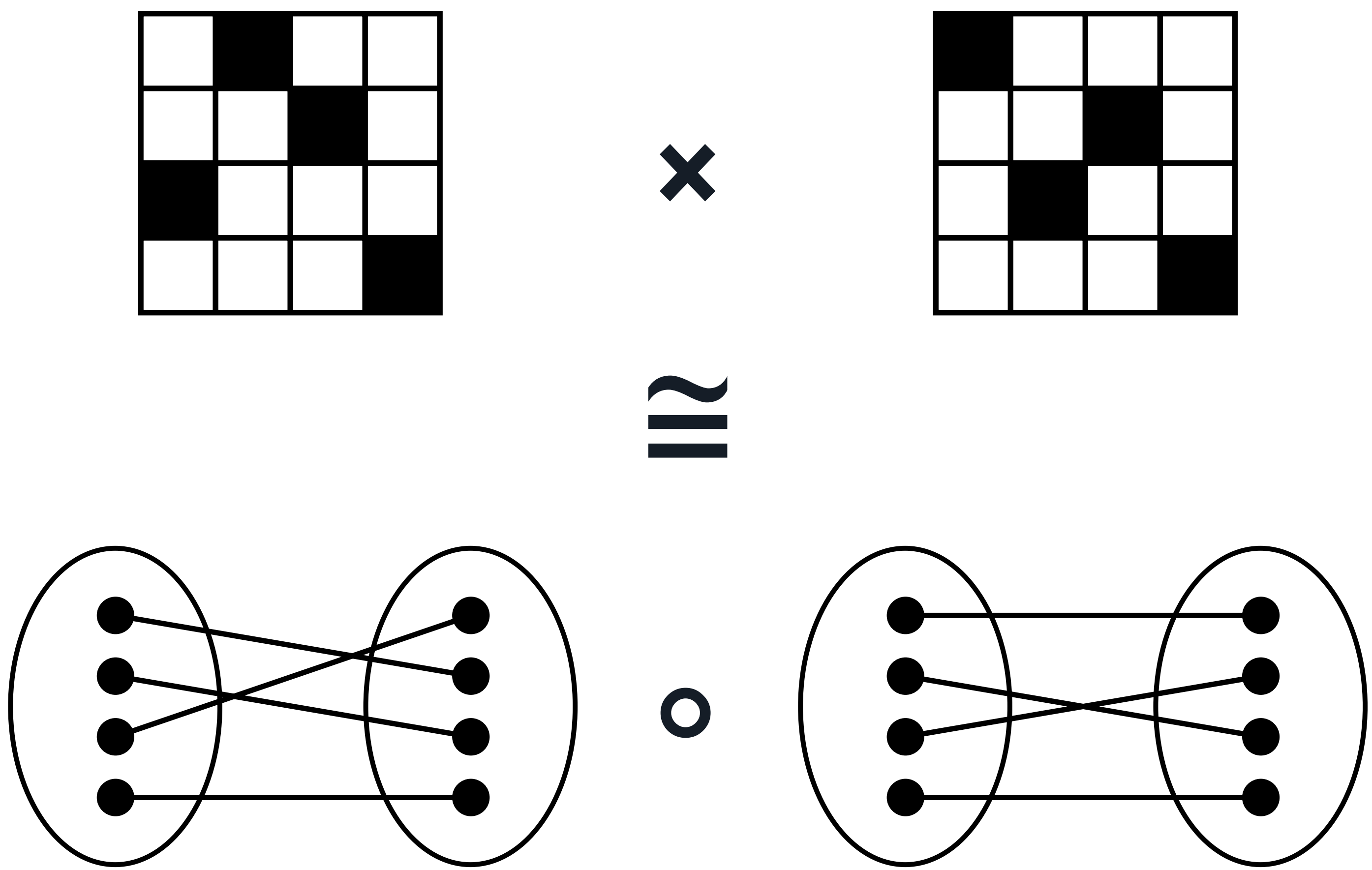
Matrix ≅ Bipartite Graph

Putting these ideas together we can see how both the boolean matrix and the bipartite graph can act as isomorphic representations of arbitrary function mappings.



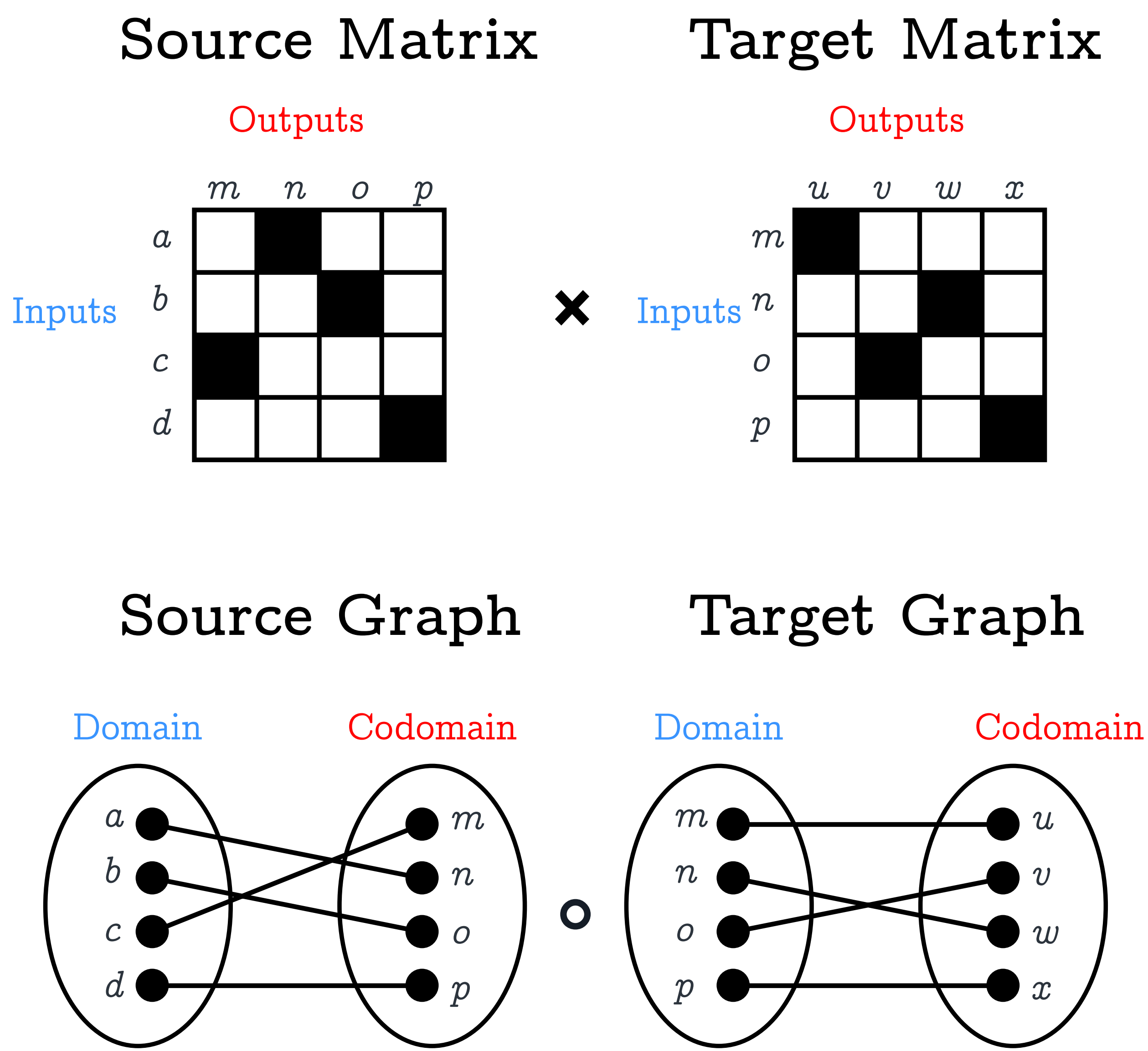
Matrix Multiplication as Function Composition

Turns out this provides a good setting to understand how matrix multiplication is function composition.



Multiplication ≅ Composition

When a matrix is thought of as representing mappings, matrix multiplication becomes equivalent to composing together two maps. The result we obtain has the inputs of source matrix mapped on to the outputs of the target matrix.

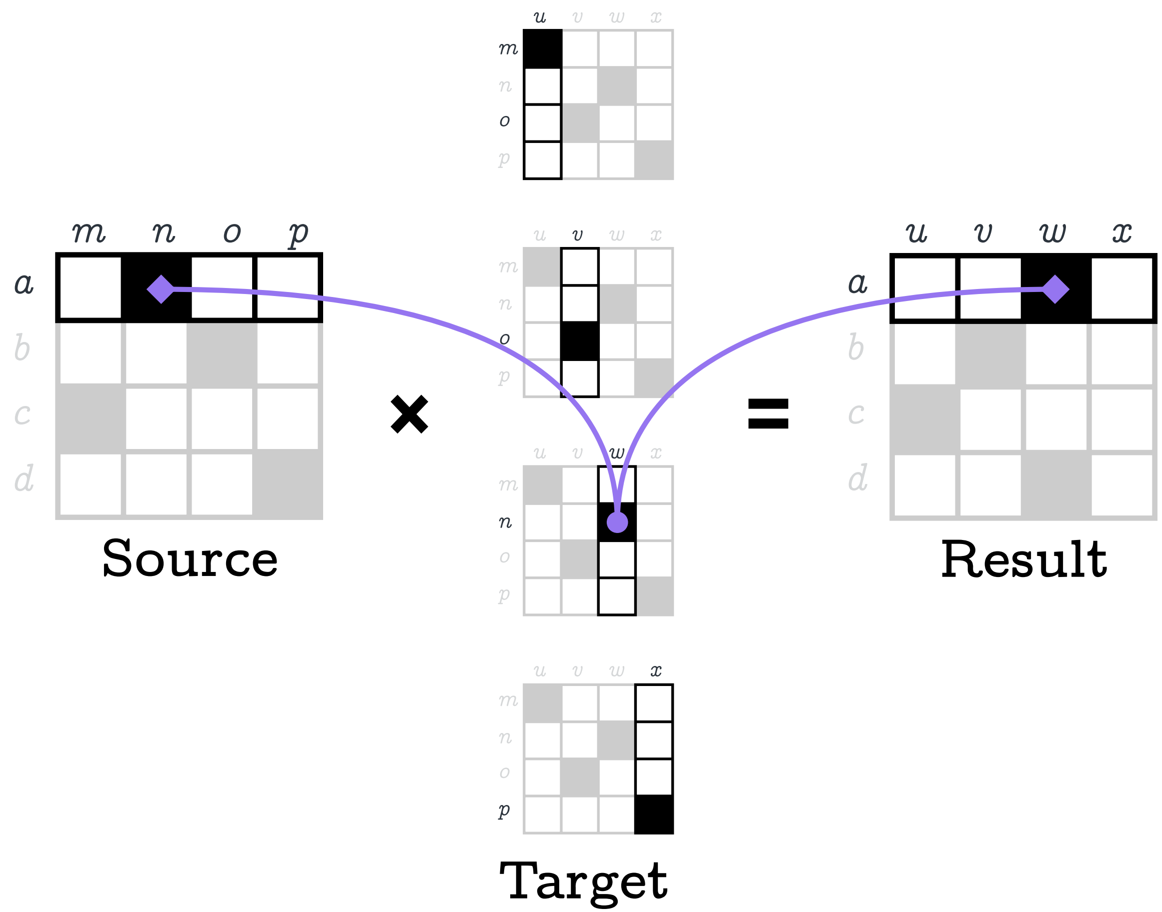


Composing Maps

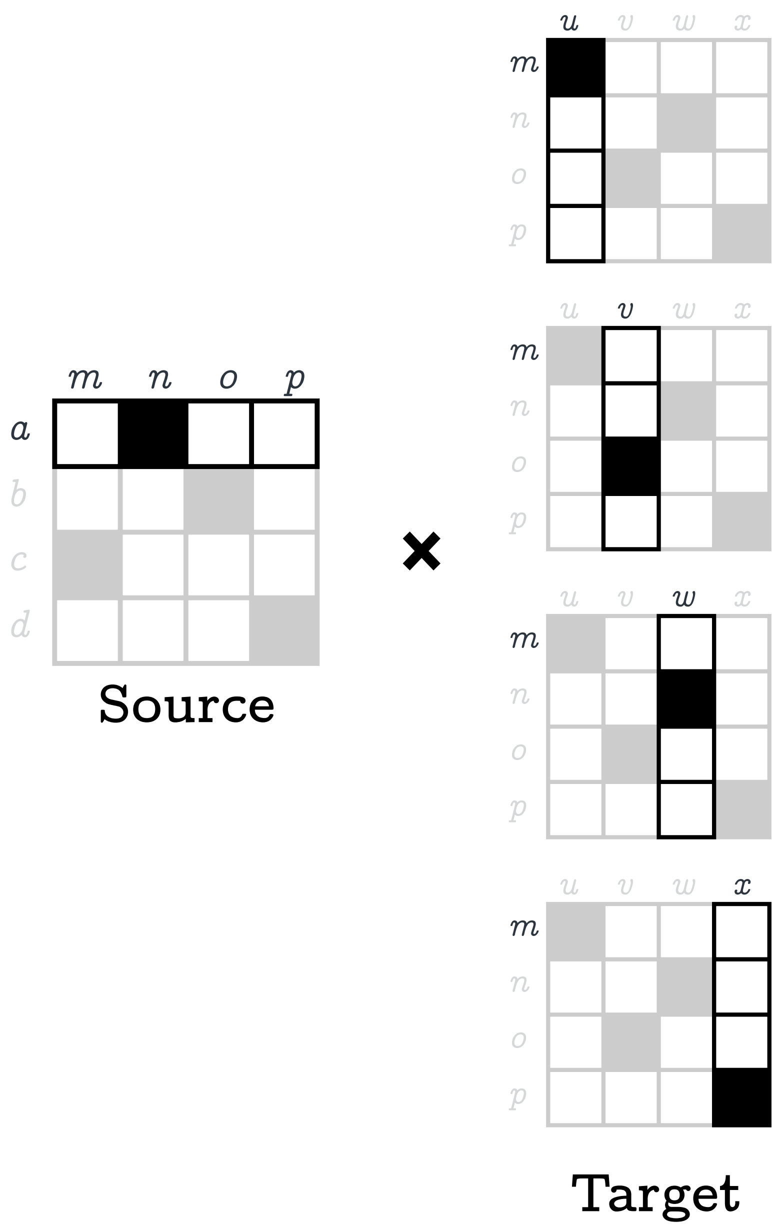
When we multiply two matrices we are in effect computing what the inputs of the source matrix map to in terms of the second matrix.

As shown here $a \mapsto n$ when multiplied with a matrix where $n \mapsto w$ gives us $a \mapsto w$.

Let us walk through the process of matrix multiplication to see how works out.



Mechanics of Matrix Multiplication I



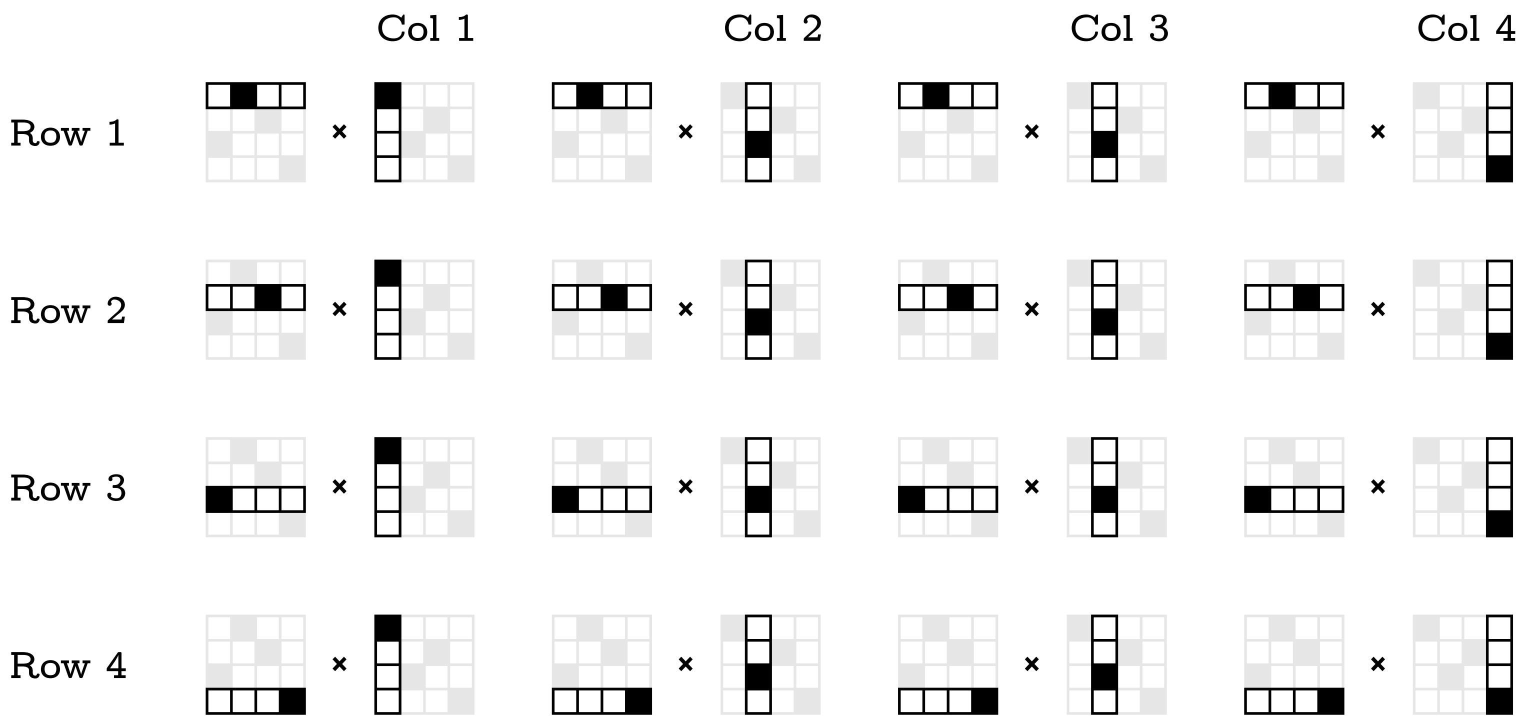
To compute matrix multiplication, we first take the input row of the source matrix and multiply it pair wise with each of the column of the target matrix.

This process transfers mapping from the row-column mappings of source matrix onto the corresponding row-column mappings of the target matrix.

Let us see how the computation achieves this.

Mechanics of Matrix Multiplication II

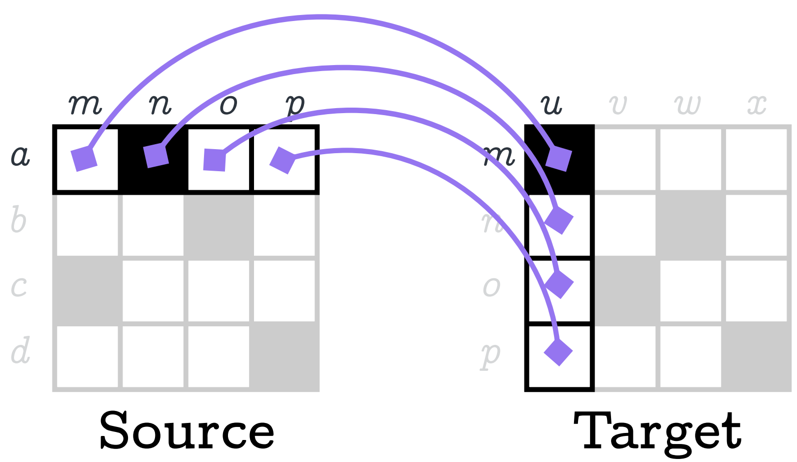
@prabros • v0.01



To multiply two matrices, each row in the source matrix is paired with each of the columns in the target matrix. This means to multiply two 4×4 matrices, each of the 4 rows in the source has to be multiplied with 4 columns of the target, resulting in a total of 16 multiplications.

Mechanics of Matrix Multiplication III

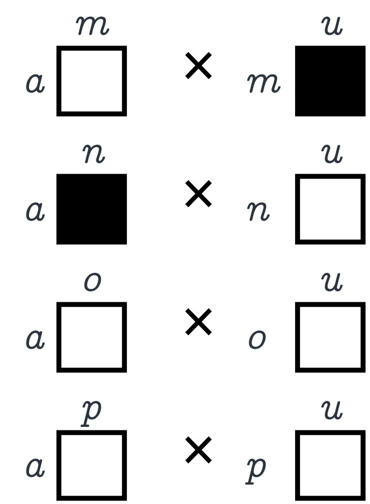
Let us see how to multiply a row of the source matrix with a column of the target matrix. This computation will become the first spot in the first row of the resulting matrix.



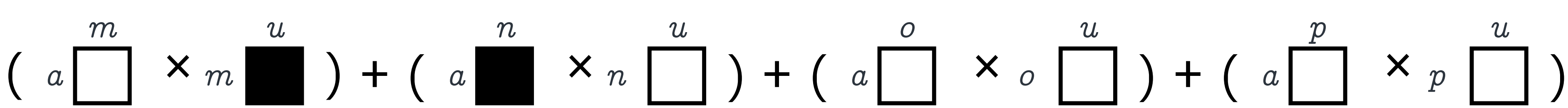
To multiply a row by a column, we have to pair the first element of the row with the first element of the column, second with the second and so on.

Mechanics of Matrix Multiplication IV

In our case, for the first row and column, it will generate the following multiplications.



These products are then to be added together to generate the first element in the result matrix.



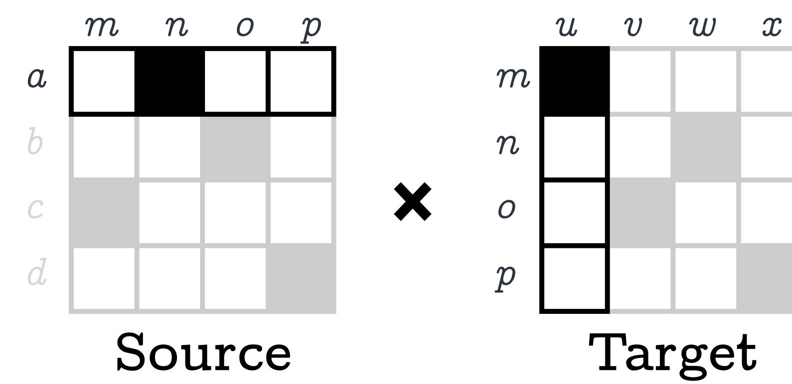
Rules of Combination

To generate our results, we have the following rules for combining the spots. If you are familiar with ideas in logic/set theory, you can easily see that the $+$ operation corresponds to the OR/Union operator and $\times$ maps to the AND/Intersection operator respectively.

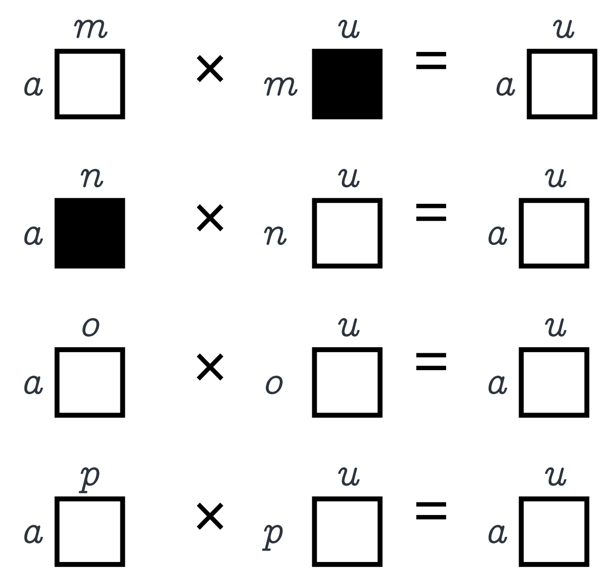
Multiplication		Addition	
□ × □ = □	$0 \times 0 = 0$	□ + □ = □	$0 + 0 = 0$
□ × ■ = □	$0 \times 1 = 0$	□ + ■ = ■	$0 + 1 = 1$
■ × □ = □	$1 \times 0 = 0$	■ + □ = ■	$1 + 0 = 1$
■ × ■ = ■	$1 \times 1 = 1$	■ + ■ = ■	$1 + 1 = 1$

First Spot Multiplication

Using the multiplication rules for combining spots we pairwise multiply individual spots of the row and column.



For the first row-column iteration we get these multiplications



First Spot Addition

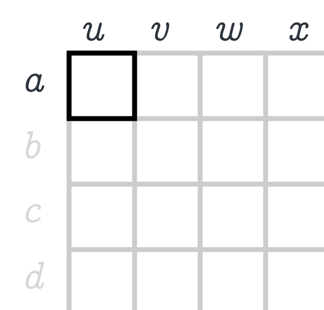
These multiplications are then combined using the + operation. Thus for the first row-column iteration we get

$$\left({}_{a}\overset{m}{\square} \times {}_{m}\overset{u}{\blacksquare} \right) + \left({}_{a}\overset{n}{\blacksquare} \times {}_{n}\overset{u}{\square} \right) + \left({}_{a}\overset{o}{\square} \times {}_{o}\overset{u}{\square} \right) + \left({}_{a}\overset{p}{\square} \times {}_{p}\overset{u}{\square} \right)$$

which reduces down into

$${}_{a}\overset{u}{\square} + {}_{a}\overset{u}{\square} + {}_{a}\overset{u}{\square} + {}_{a}\overset{u}{\square} = {}_{a}\overset{u}{\square}$$

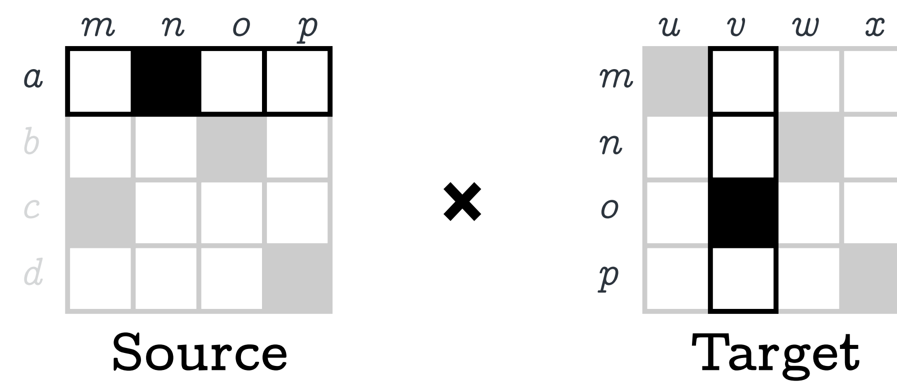
This becomes the first spot of the first row of the result which is an empty spot representing the value 0. It means that there is no connection between a and u in our result matrix.



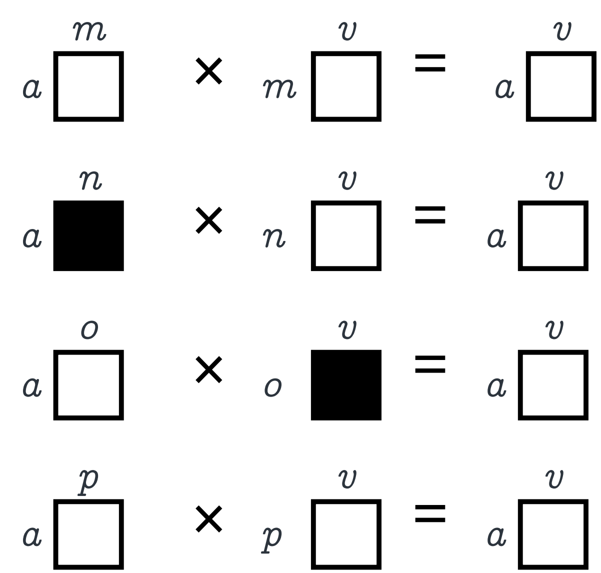
Result

Second Spot Multiplication

Now, let us walk through the second row.



And we get the following multiplications

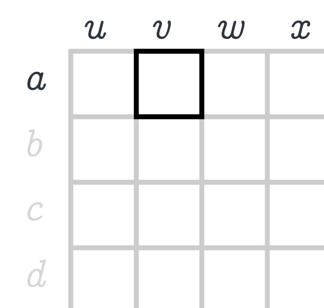


Second Spot Addition

Here too, we can't find a dark spot / match.

$$a\overset{v}{\square} + a\overset{v}{\square} + a\overset{v}{\square} + a\overset{v}{\square} = a\overset{v}{\square}$$

This compuation goes into the second spot of the first row of the result, which also becomes a blank spot like the last one.

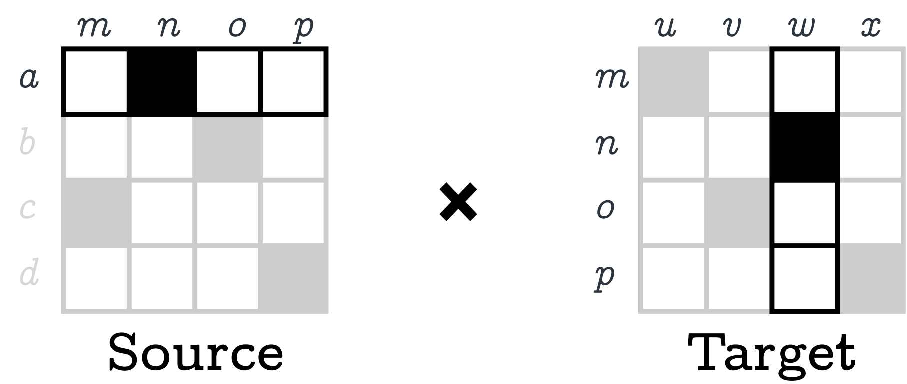


Result

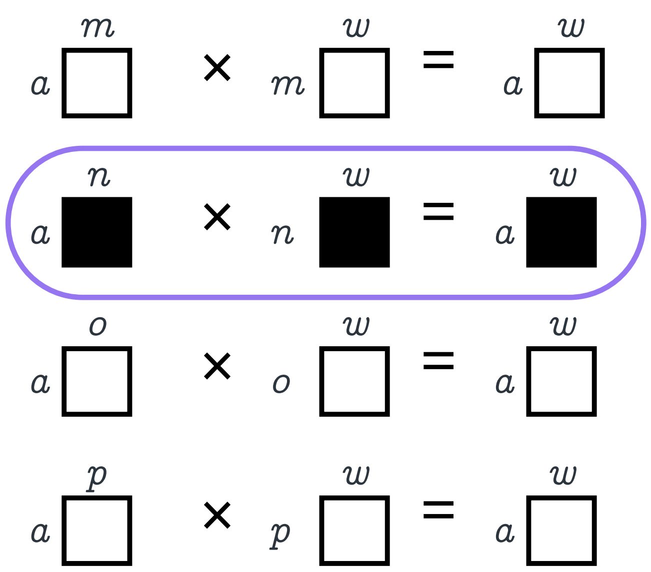
Third Spot Multiplication

@prabros ● v0.01

Let us check out what the case with third one is.



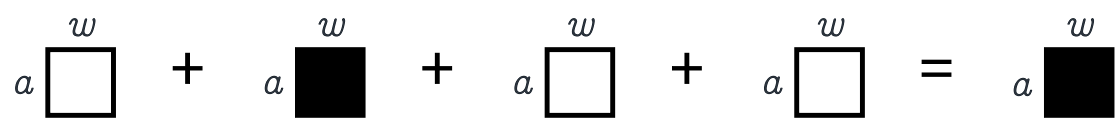
The generated multiplications are



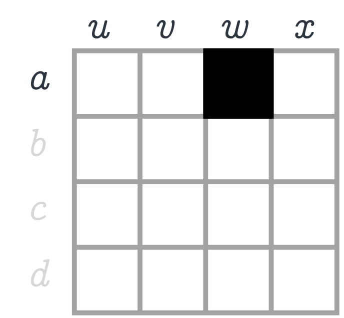
And there's a match on the second index of the first row of the source and the second index of the third column of the target!

Match Found!

And this will give us the following additions



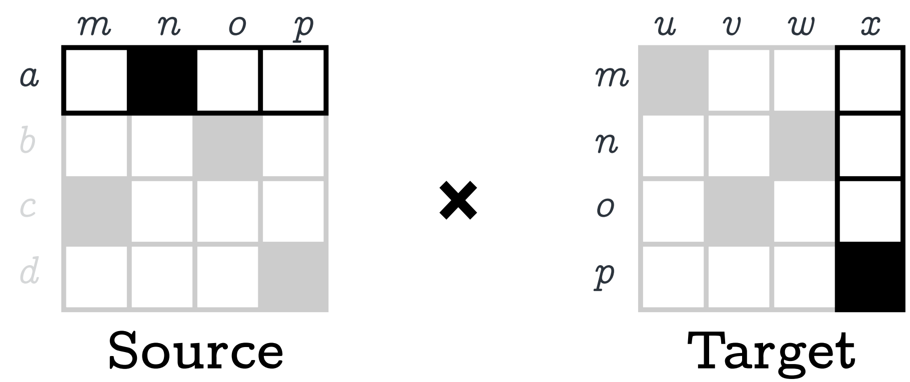
Having at least one dark spot means that on addition, this will also generate a dark spot. Thus the third spot in the first row of the resultant matrix turns out to have a match!



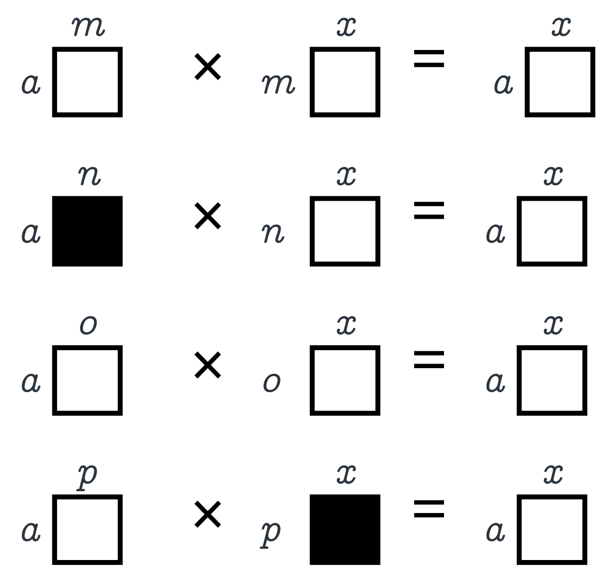
Result

Fourth Spot Multiplication

And for completion let us walk through last multiplication of first row of source matrix.



It generates these multiplications

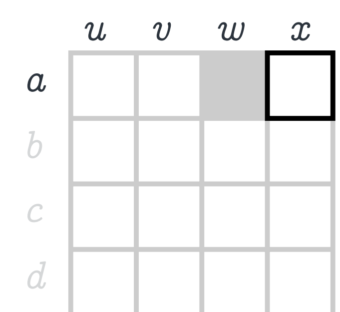


Fourth Spot Addition

And since no match was found, this will reduce to an empty spot.

$$
{}_{a}\overset{x}{\boxed{}} + {}_{a}\overset{x}{\boxed{}} + {}_{a}\overset{x}{\boxed{}} + {}_{a}\overset{x}{\boxed{}} = {}_{a}\overset{x}{\boxed{}}
$$

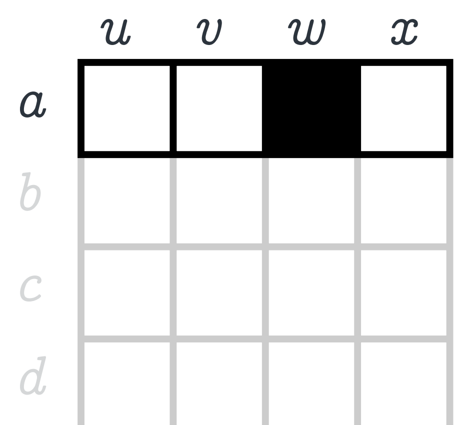
which becomes the fourth spot in the first row.



Result

First Row Result

So we computed our result for the first row:

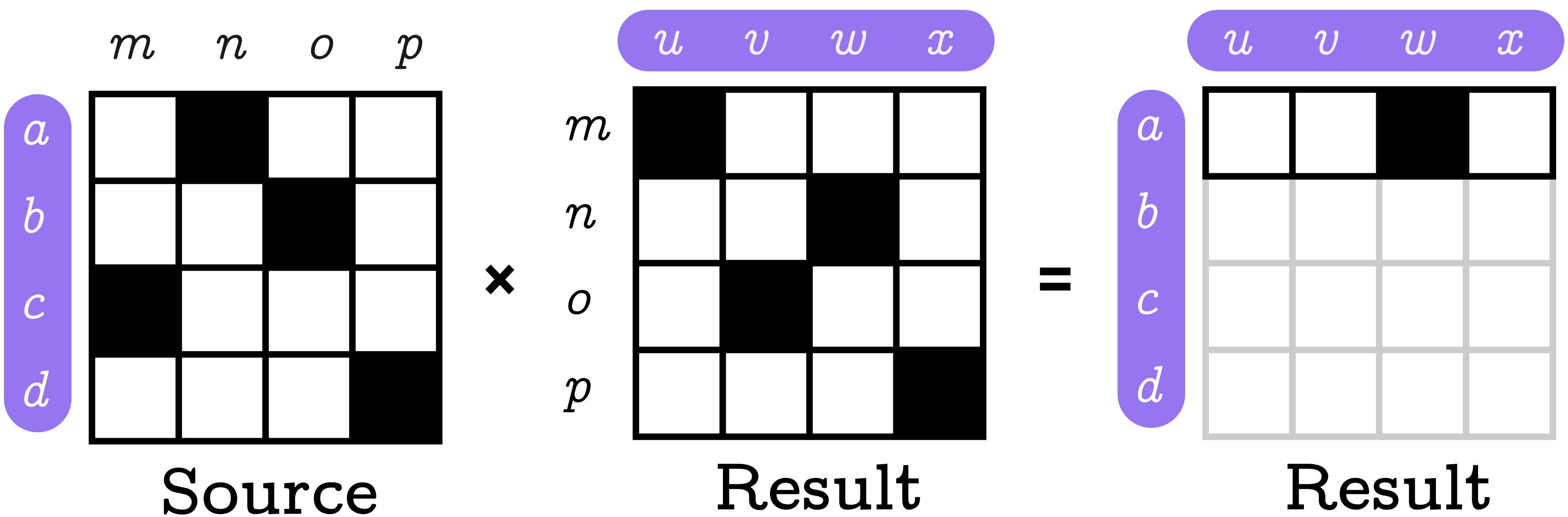


Turns out, the result has a single dark spot and the rest empty. It follows the characteristic of a logical matrix of a function that an input only ever maps to a single output. This property will be maintained throughout the next matrix multiplications.

Basis Shift

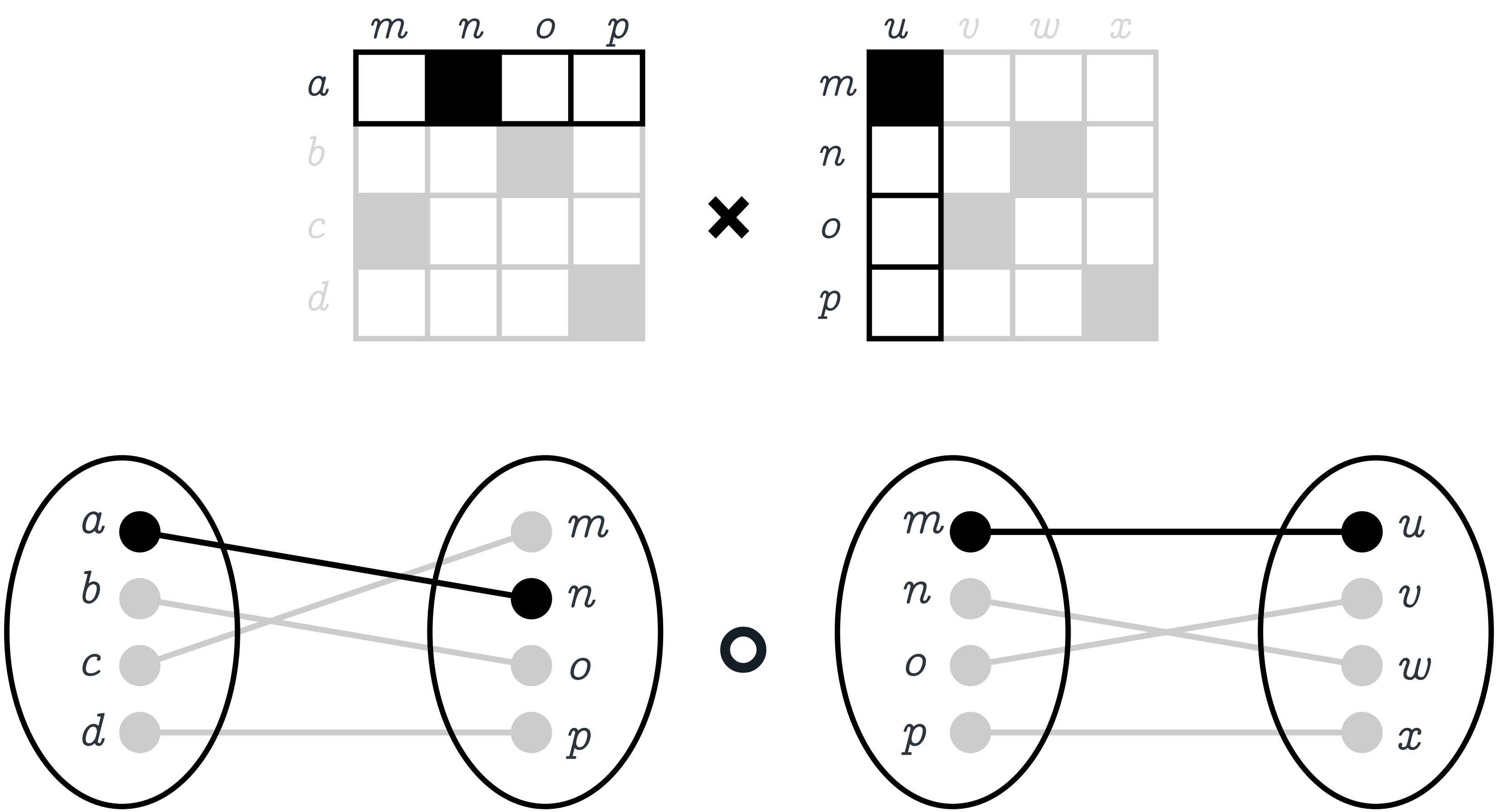
@prabros ● v0.01

Notice that the result matrix has as index $\{\ a,\ b,\ c,\ d\ \}$ for the rows, which correspond to the rows of the source matrix and $\{\ u,\ v,\ w,\ x\ \}$ as columns which correspond to columns of the target matrix. So in effect we are creating a new mapping from the inputs of the source matrix to the outputs of the target matrix!

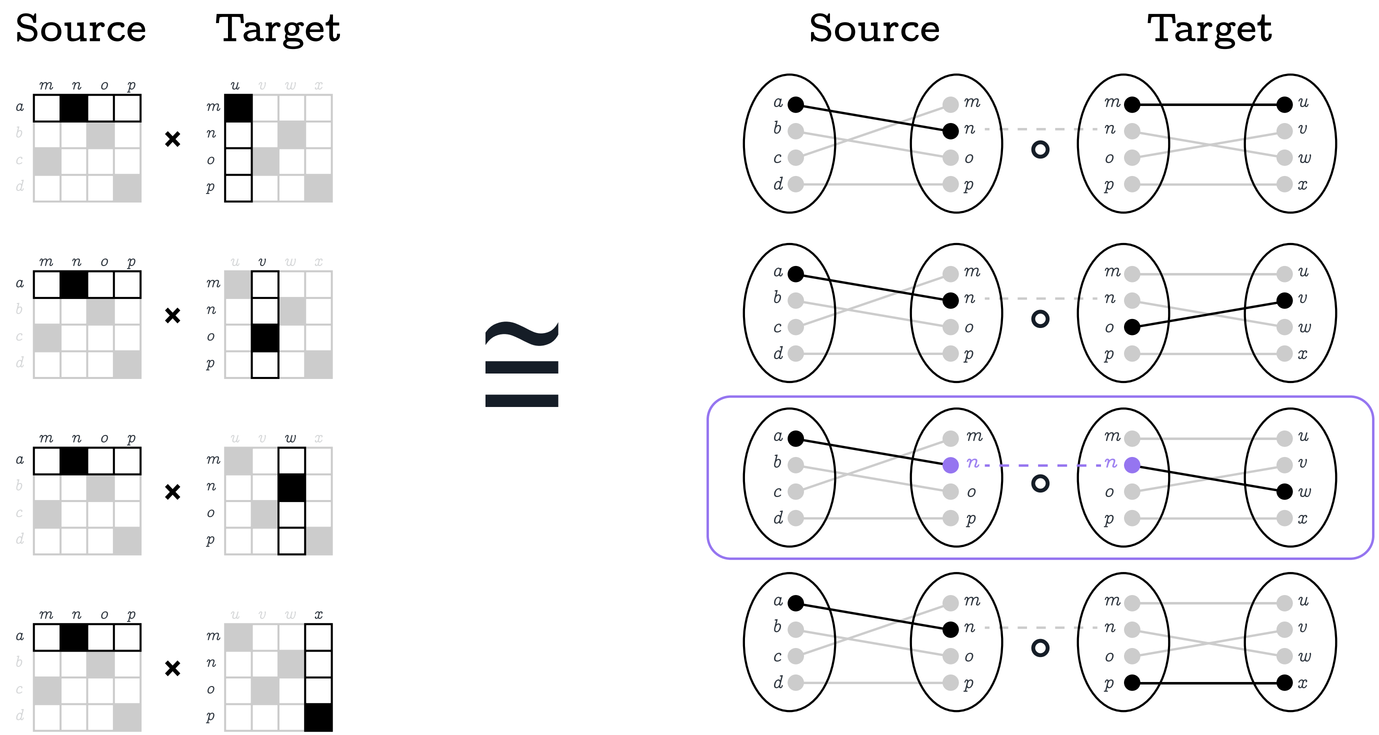


Computing with graphs

Now it is time for us to juxtapose matrix computations with functional bipartite graphs and study them comparatively.

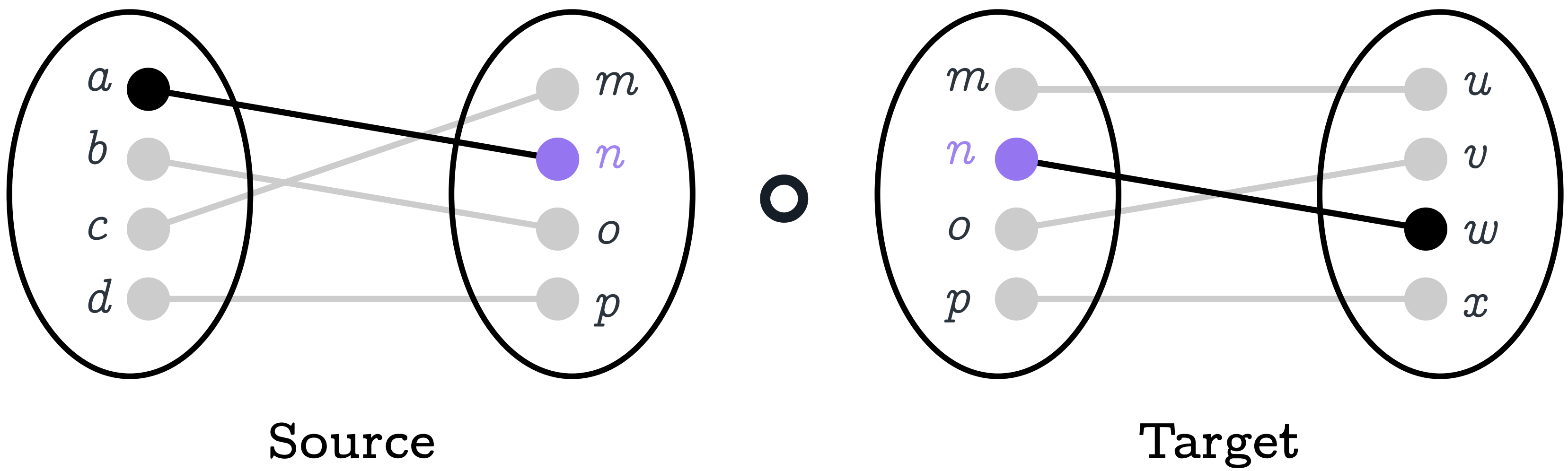


Matrix multiplication with graphs



Multiplication of a row with columns in matrix form is equivalent in bipartite graph representation to seeing where the output of the codomain of the source graph connects to among the inputs of the domain of the target graph. For the previous matrix multiplication computing first row of result matrix, output of source graph $a \mapsto n$ matches with the mapping $n \mapsto w$ in the target graph.

Excluding the Common Middle

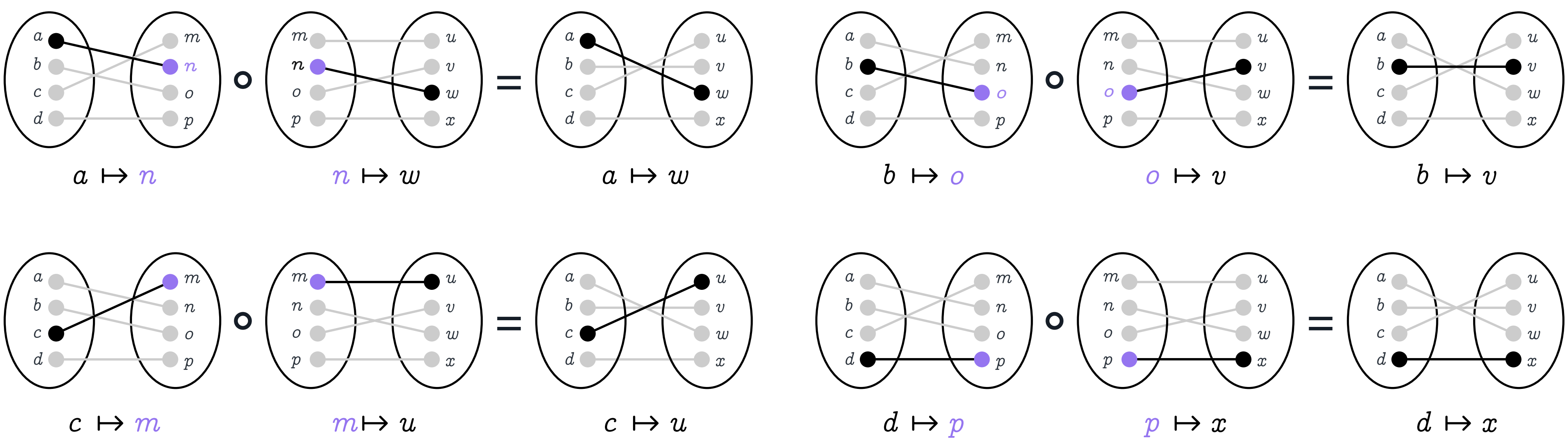


Once we find the common connection, we generate our result by drawing an edge between input of the source graph and corresponding output of the target graph eliding the intermediary points.

In the above example this effectively takes us from: $a \mapsto n \circ n \mapsto w$. Eliding the common intermediary n, we get: $a \mapsto w$. Thus we get the result where input a of the domain of the source function is mapped to output w in the codomain of the target function via n.

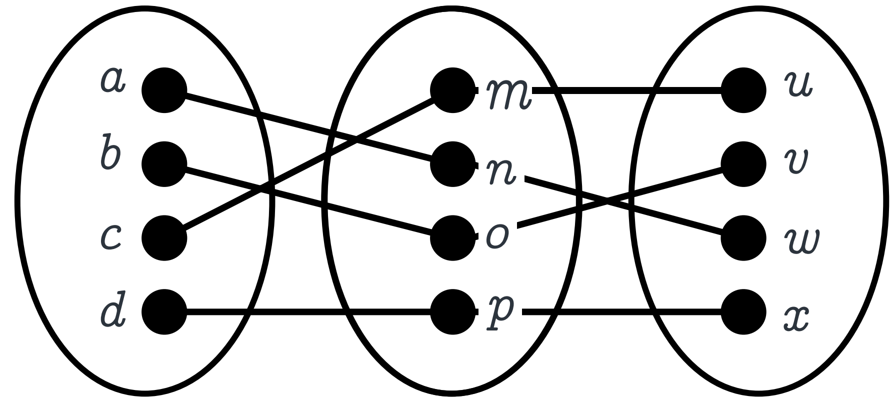
Swift composition with graphs

In effect, we are seeking the mutual connections that exist between codomain of source graph and domain of target graph — we look at source outputs and then check what they map to in the target graph. Once we understand where they lead up to, we connect the inputs of source with these outputs in the target. By drawing such an edge, we generate a new graph that "hops" from the inputs of the source to outputs of the target graph. This process generates the composition!



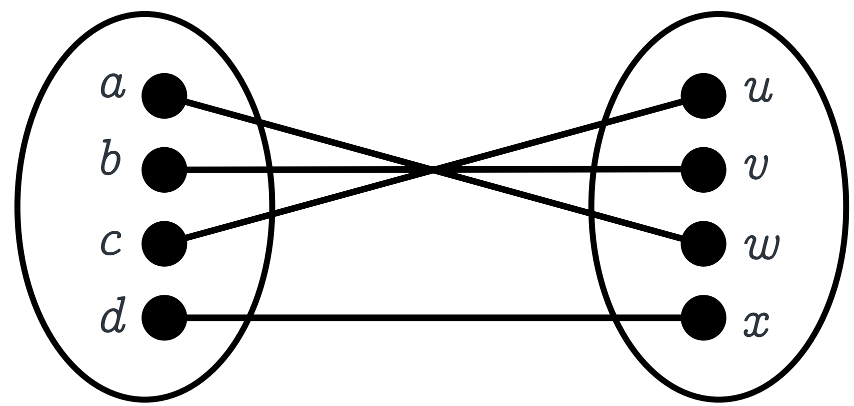
Composition ≅ Graph Hop

We obtain the result by juxtaposing two graphs and unifying the codomain of source and domain of target. With this representation, linking inputs of source to outputs of target via the intermediary set gives us the composition.



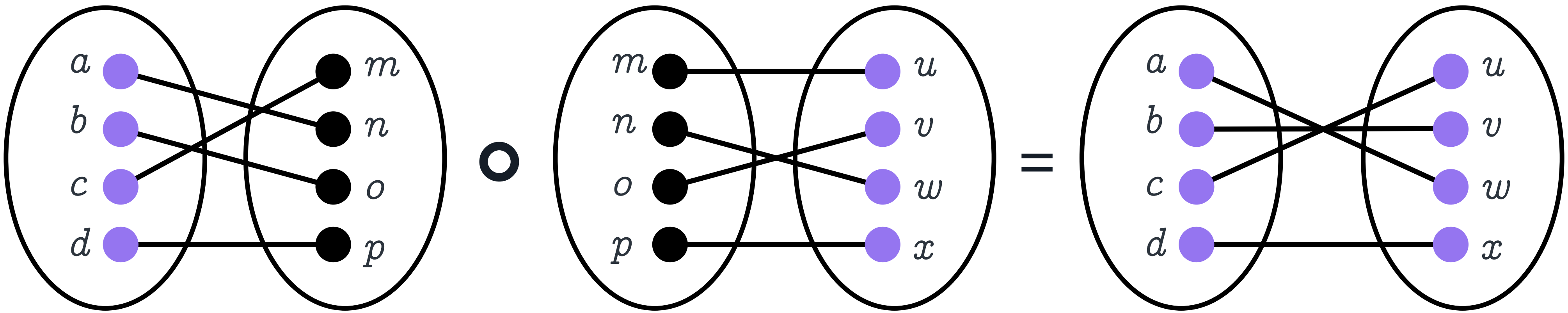
Graph composition

To compute the end result, we exclude the intermediary set and obtain a relation which maps the input of the first graph to the outputs of the second graph.



Resulting graph after hop

Graph Hop Basis Shift

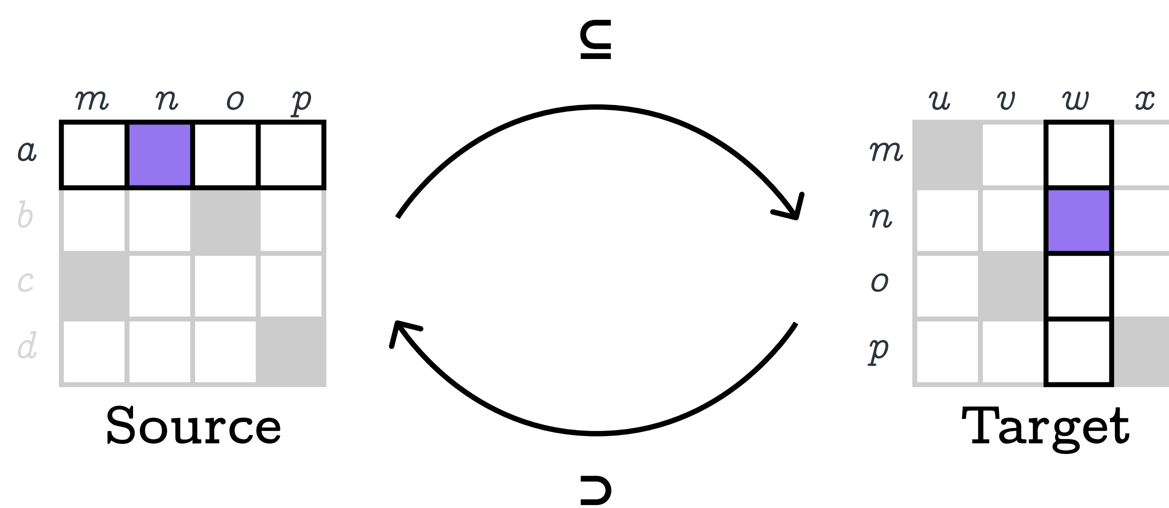


If you observe, what is happening here is a graph hop! Matrix multiplication is transitively doing a graph hop that jumps from inputs of source to outputs of target. Connecting the source inputs to the target outputs give us this hop, which as noted in the case of the matrix, gives us source inputs mapped to the target outputs in terms of the target inputs.

Faster Matrix Multiplication

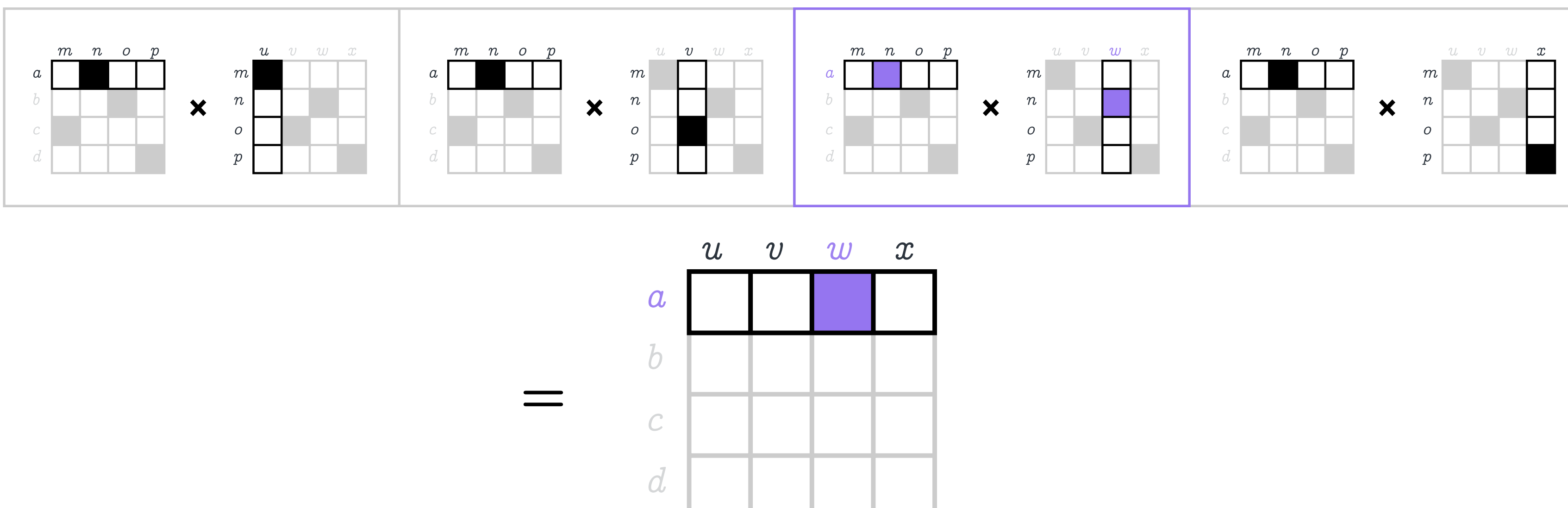
We saw how traditional matrix multiplication works, but that was a rather intricate process. It gets daunting when the matrix grows in complexity. But there turns out to be a neat economic way to just compute the result by eye!

The graph hop idea ushers us towards an economic way to compute the result. We can do this visually and quickly write down the matrix. Let us see how!



Local Insight for Computation

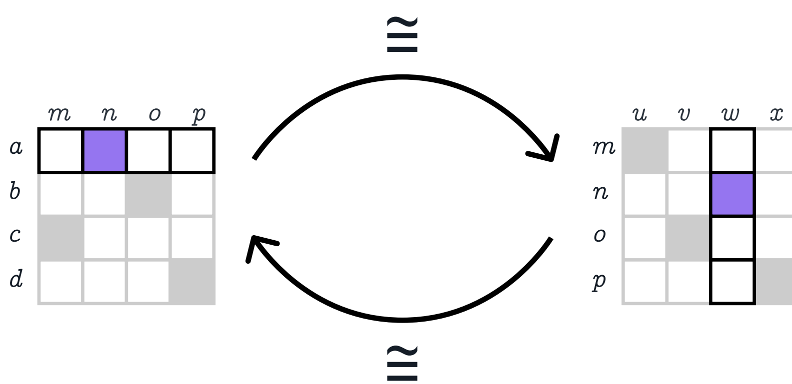
An insight for computing matrices quickly: the result matrix will have a dark spot in the mapping between those indices where the same index of the row of source matrix and the column of output matrix share a dark spot.



In our case, for first row-column multiplication, we can see that the first row of source labelled a and second column of target labelled w share a dark spot on the same second index, this means $a \mapsto w$ will be connected in the result.

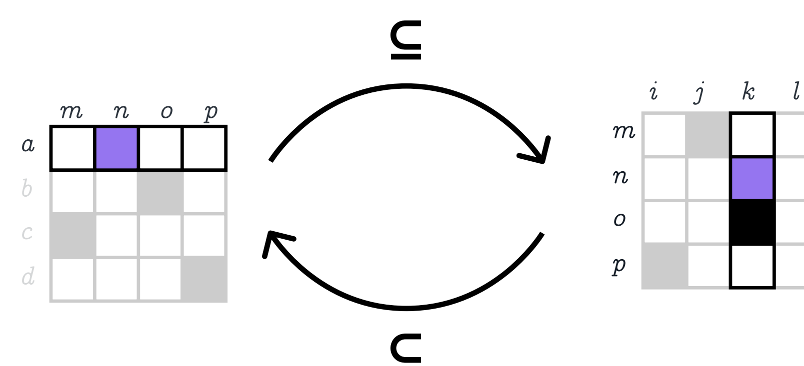
Global Insight for Computation

That was a "local" way to see matrix multiplication. Turns out there is a global way to interpret it!



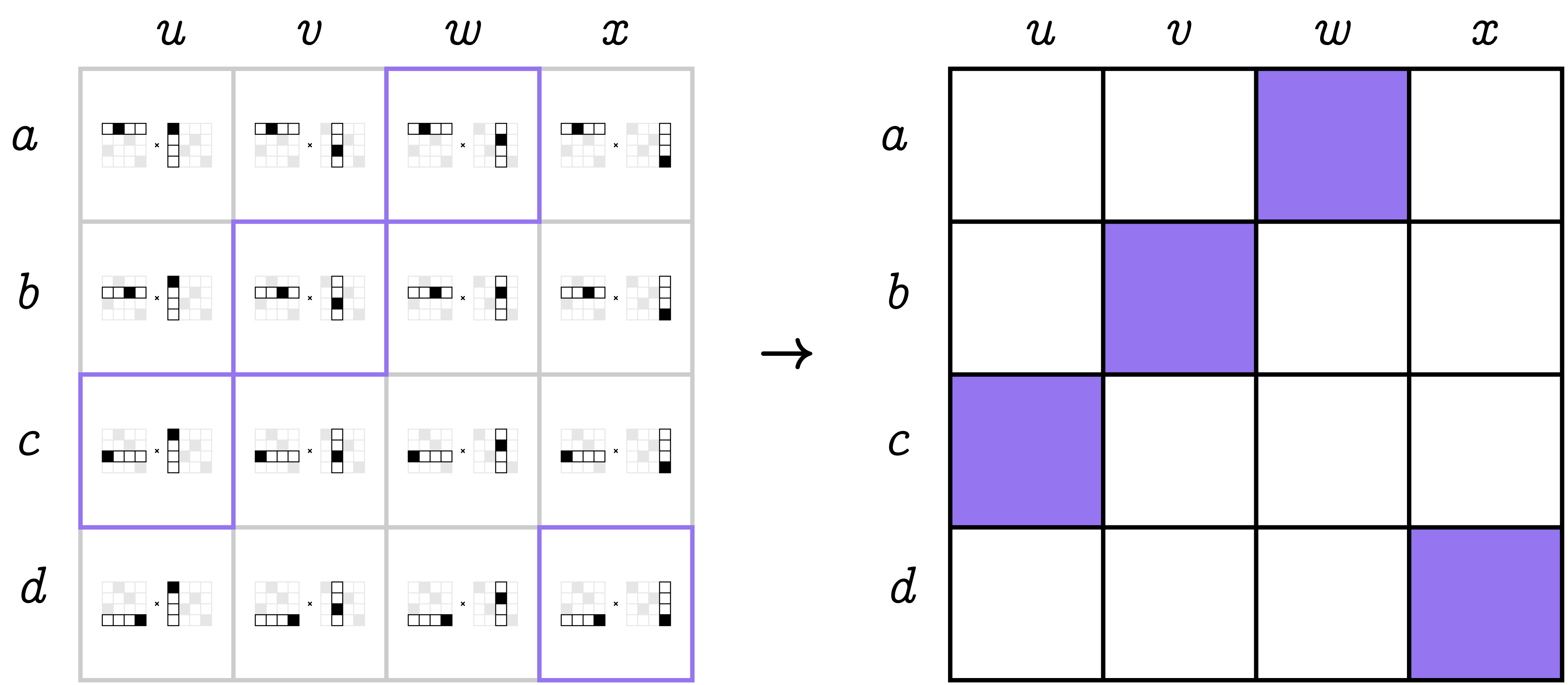
Given both the source columns and target rows are in the same order, a match is found when a target column is a rotated copy of the source row we are multiplying it with. This means on composing we are looking for a congruence!

There's a small wrinkle that if the function turns out to have more than one dark spot in the same column, we look for an inclusion of the row in the column rather than a pure congruence.



Fast Computations using Symmetry

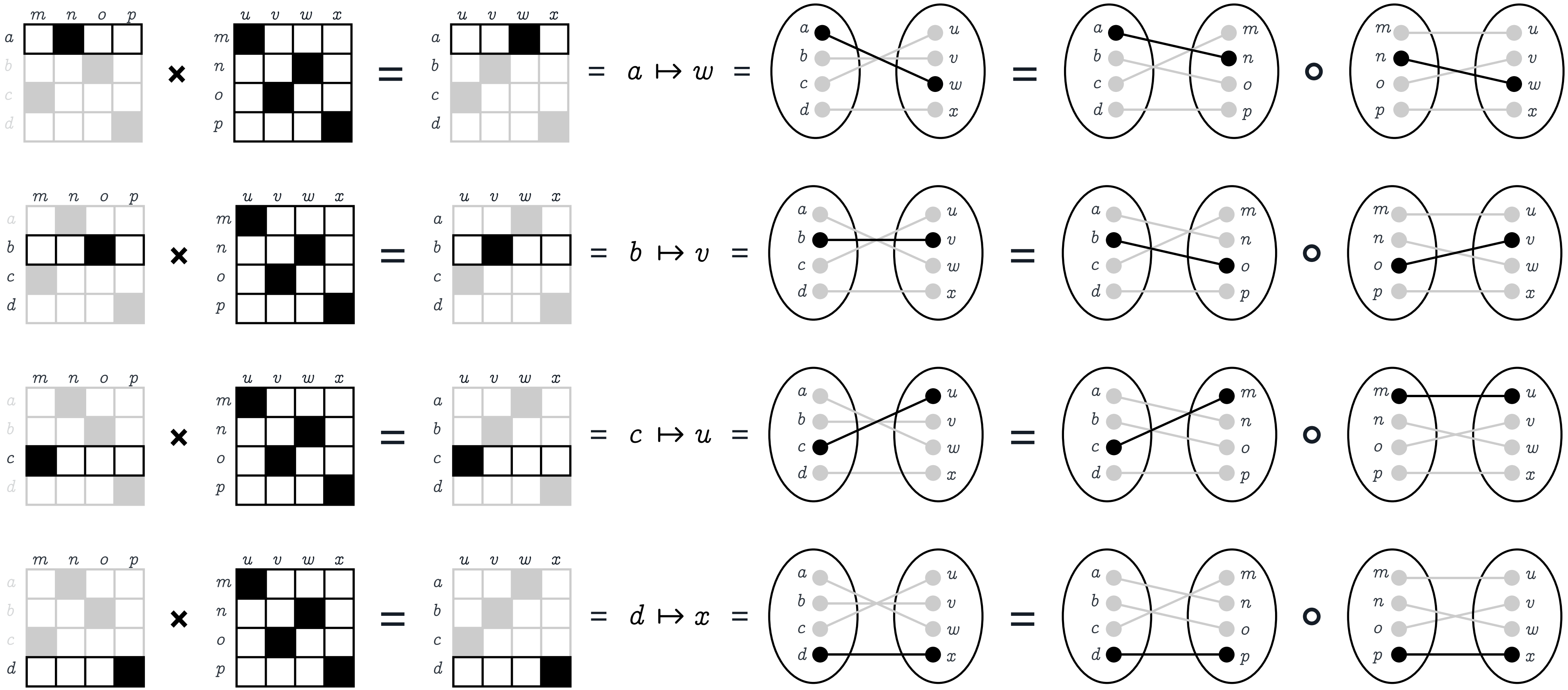
Using this insight lets us outline the spots where such a symmetry is found. This approach is comparable to lookup tables in computer programming, where we just “look” at the computations done ahead of time and draw the result.



If we mark these selections as a connection, our computation is done!

All Together Now

Now let us see both the matrix and graph representations side by side. With the multiplication/composition, we are seeking what each output of the source matrix/graph map to in the target matrix/graph.



Final Result

Putting these together, this is our final result