2-Trees

[Basis] K_2 is a 2-tree.

[Inductive step] If G is a 2-tree and (u, v) is an edge of G then

$G + w + (u,w) + (v,w)$ is a 2-tree.

A partial 2-tree is any subgraph of a 2-tree.

Goal: Count the number of spanning trees of a 2-tree by stripping off 2-leaves.

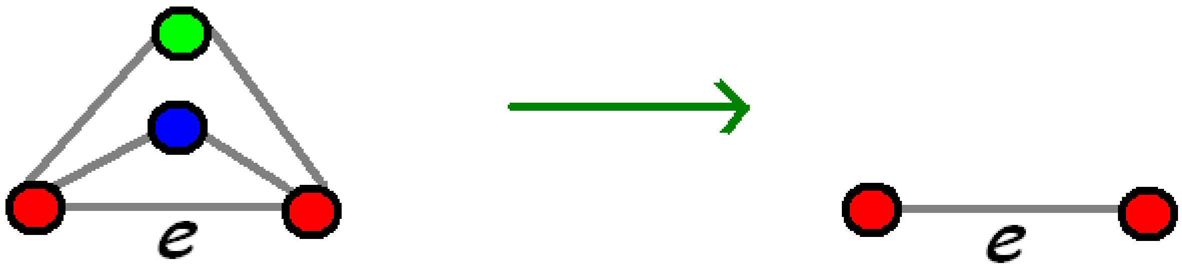
Two variables per edge:

dc((u, v)) = the number of ways to have a 2-component forest selected from the subgraph reduced down onto (u,v) so that u and v are in different components.

sc((u, v)) = the number of ways to have a 1-component forest selected from the subgraph reduced down onto (u,v)

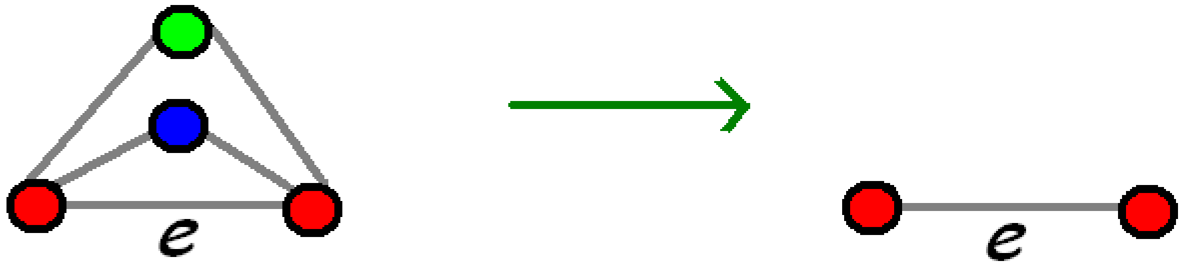
(u and v are in the same component).

What should dc(e) and sc(e) be after this reduction?

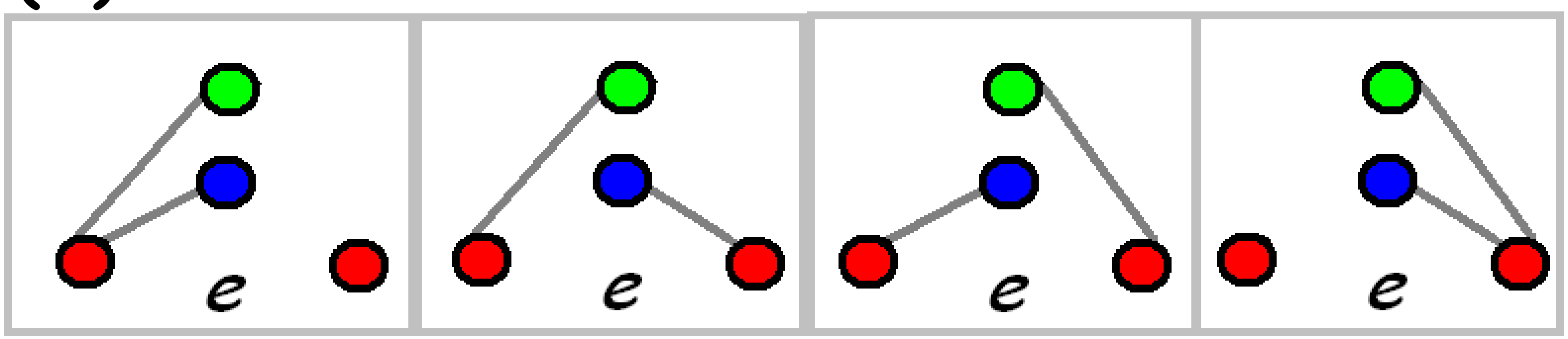


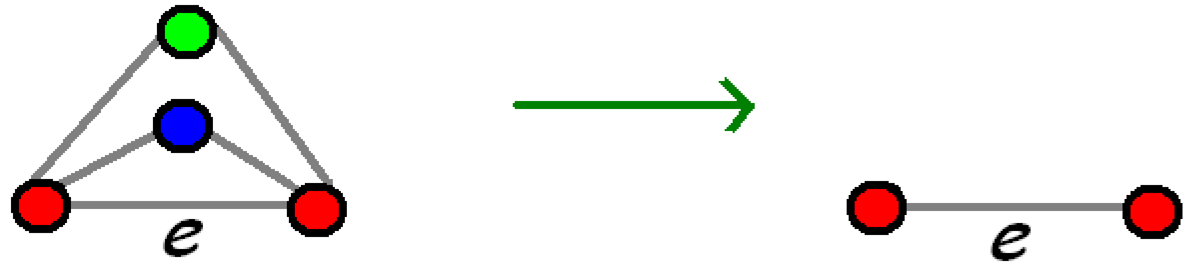
dc((u, v)) = the number of ways to have a 2-component forest selected from the subgraph reduced down onto (u,v) so that u and v are in different components.

sc((u, v)) = the number of ways to have a 1-component forest selected from the subgraph reduced down onto (u,v)

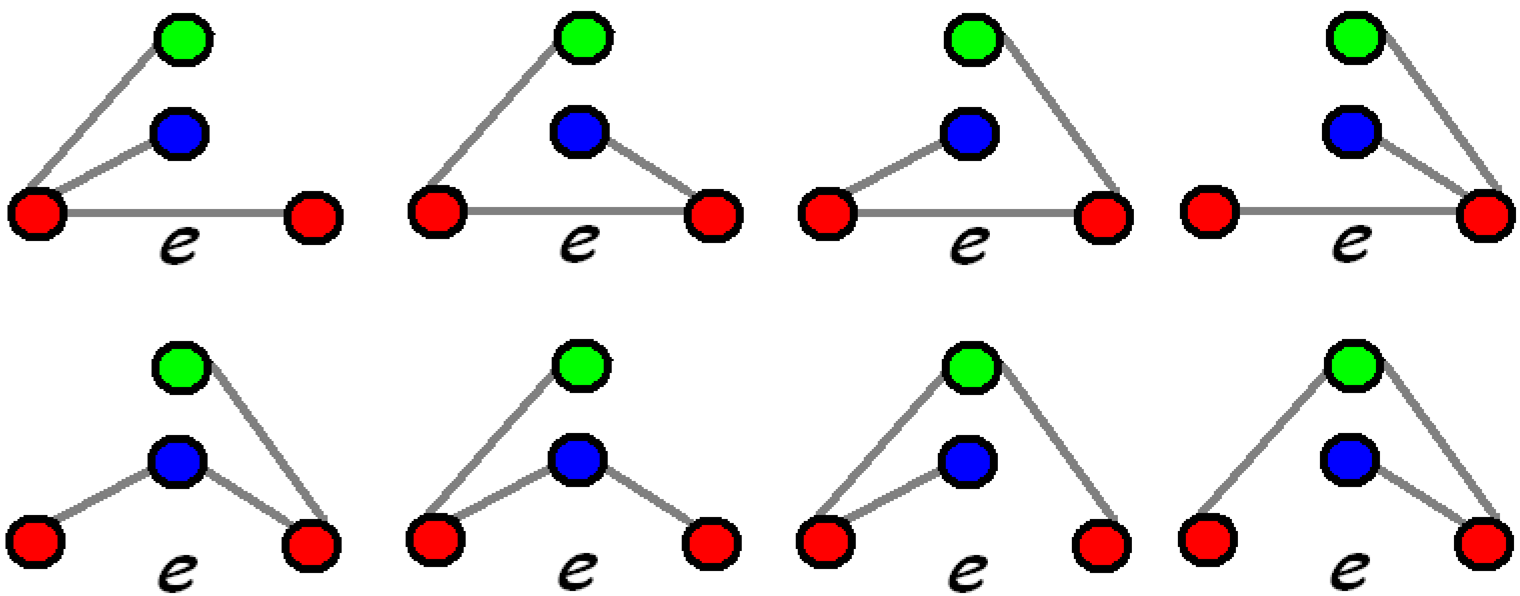


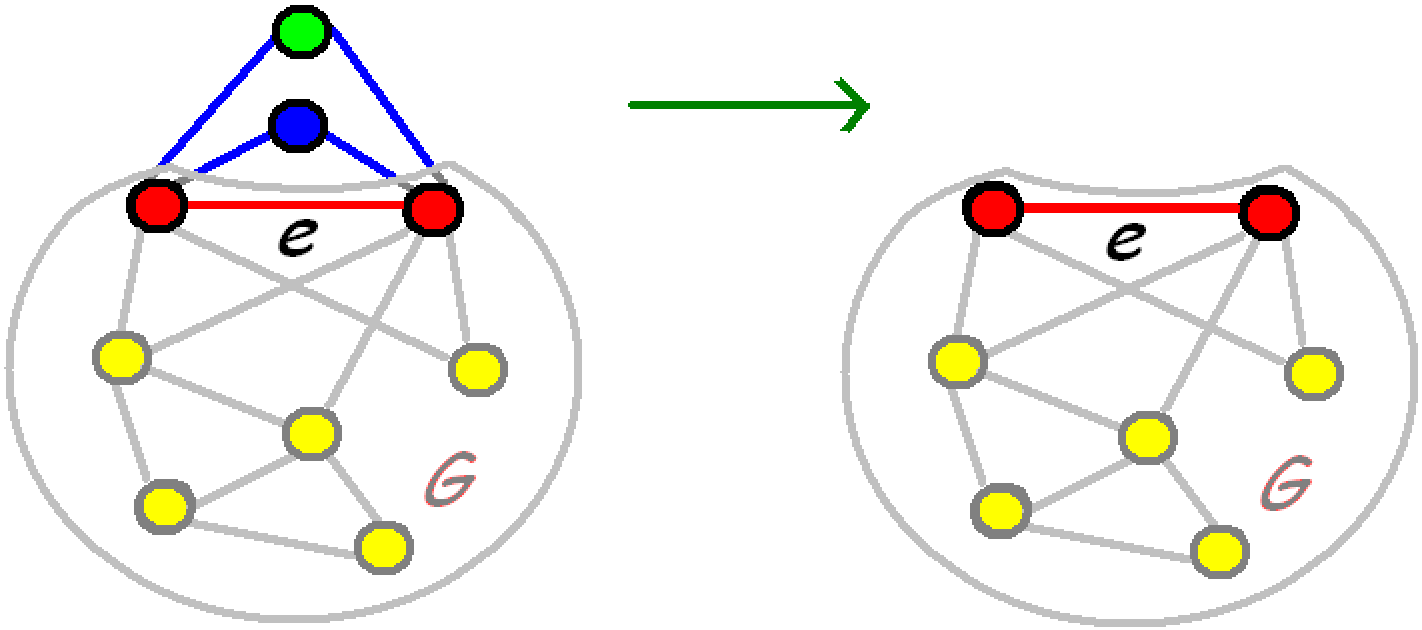
dc(e)= 4:





$sc(e) = 8$:





This reduction could be happening in a bigger setting.

Initialization:

For all edges (u, v):

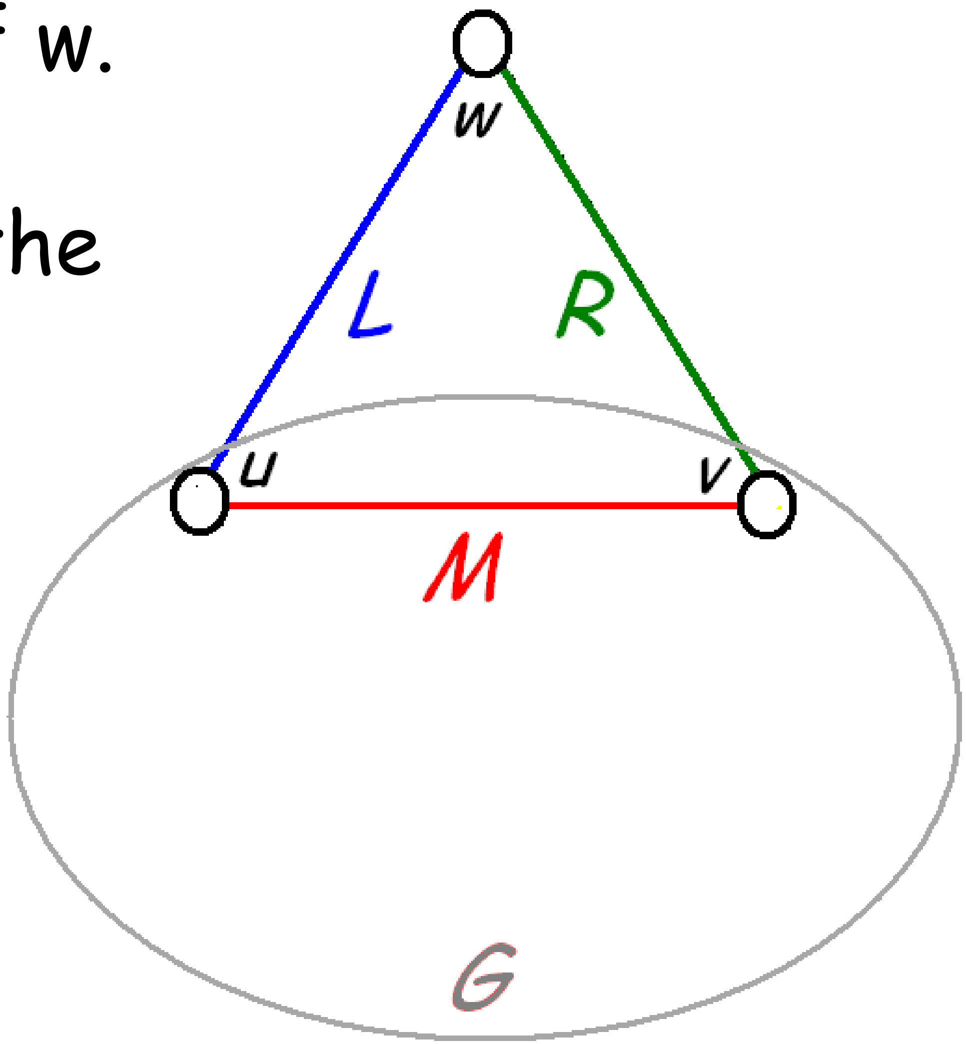
dc((u,v))= 1

sc((u,v))= 1

To handle partial k-trees, if (u,v) was an edge of the 2-tree which was deleted then you can initialize sc((u,v))= 0.

The general step:

Delete a 2-leaf w.
This means we remove w and the edges L and R:



What should the update formulas be for M?

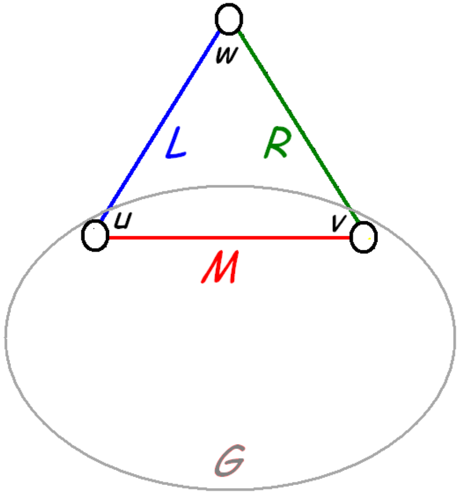
Notational convention to avoid confusion between the old/new formulas for M:

We have variables

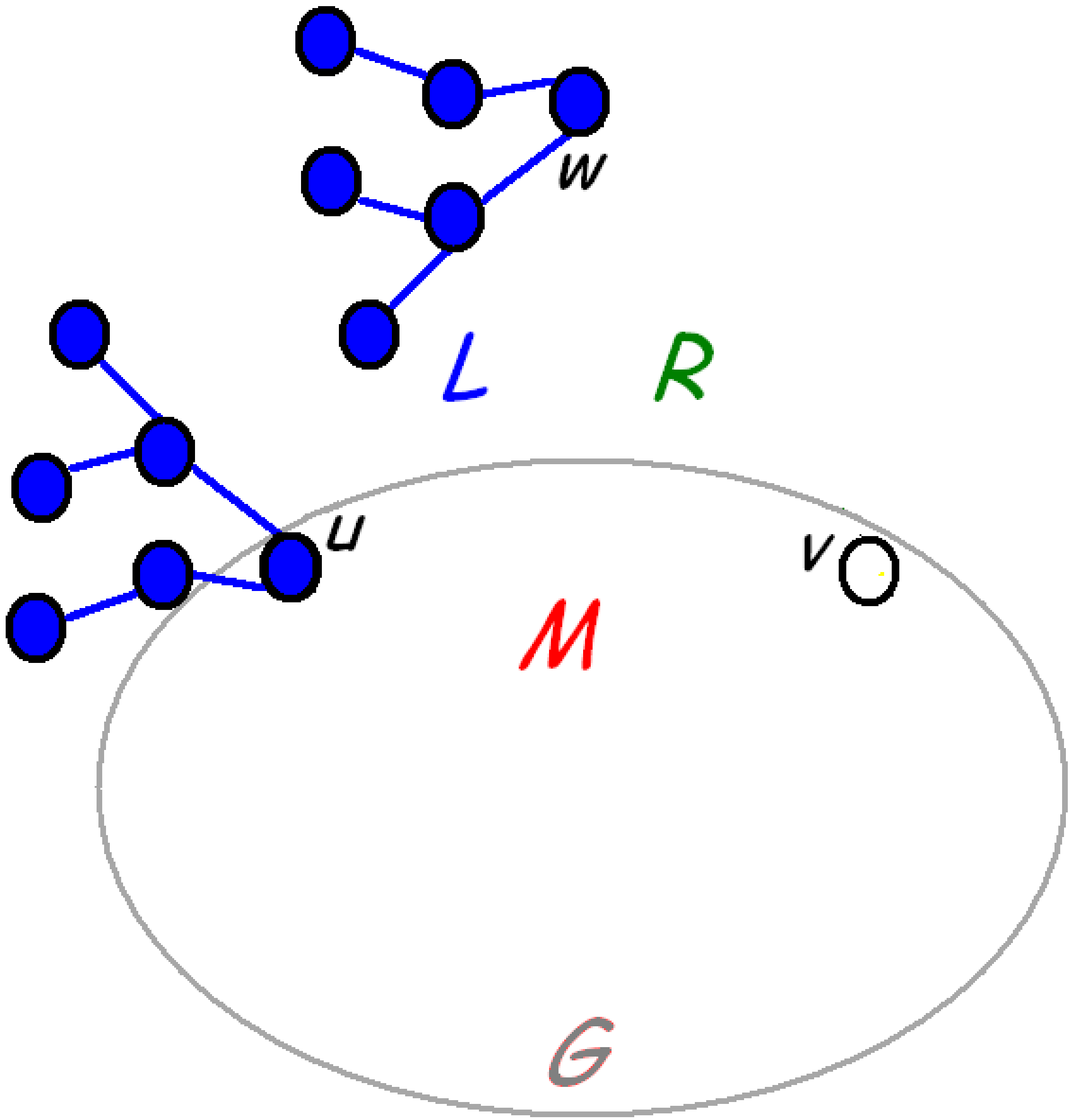
dc(L), sc(L), dc(R), sc(R), dc(M), sc(M)

The new values we compute for M will be denoted by

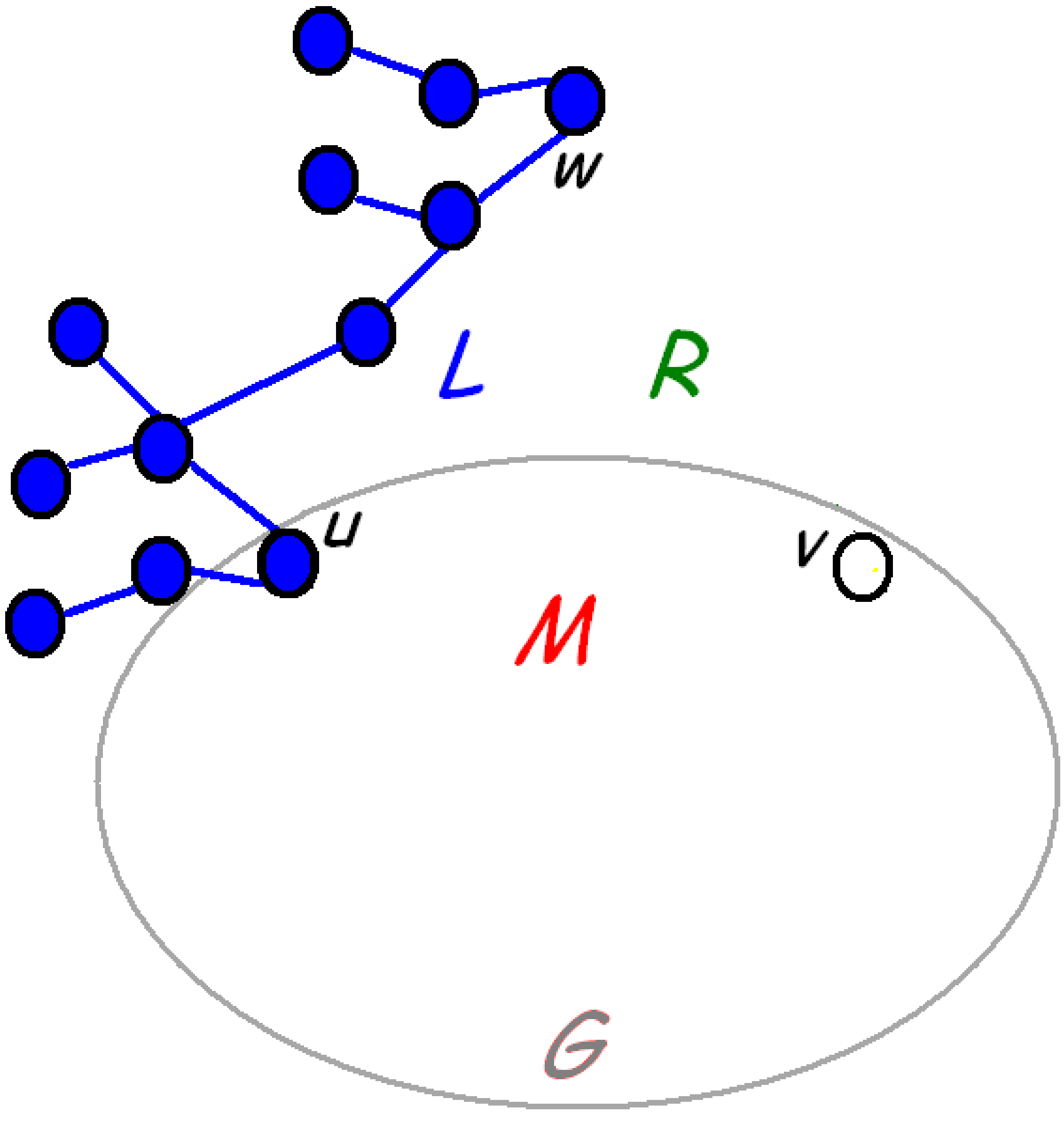
dc(M'), sc(M')



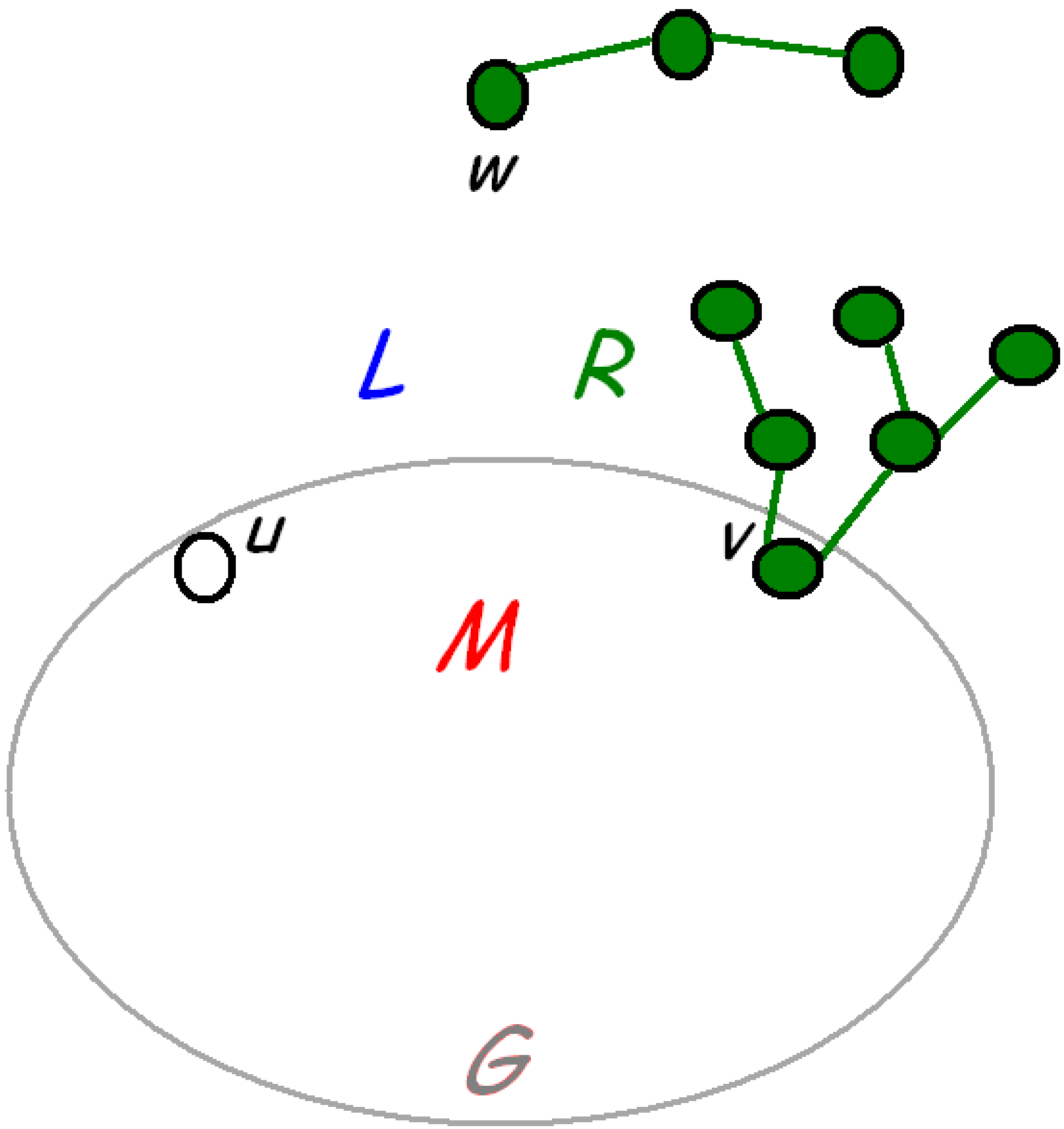
dc(L):



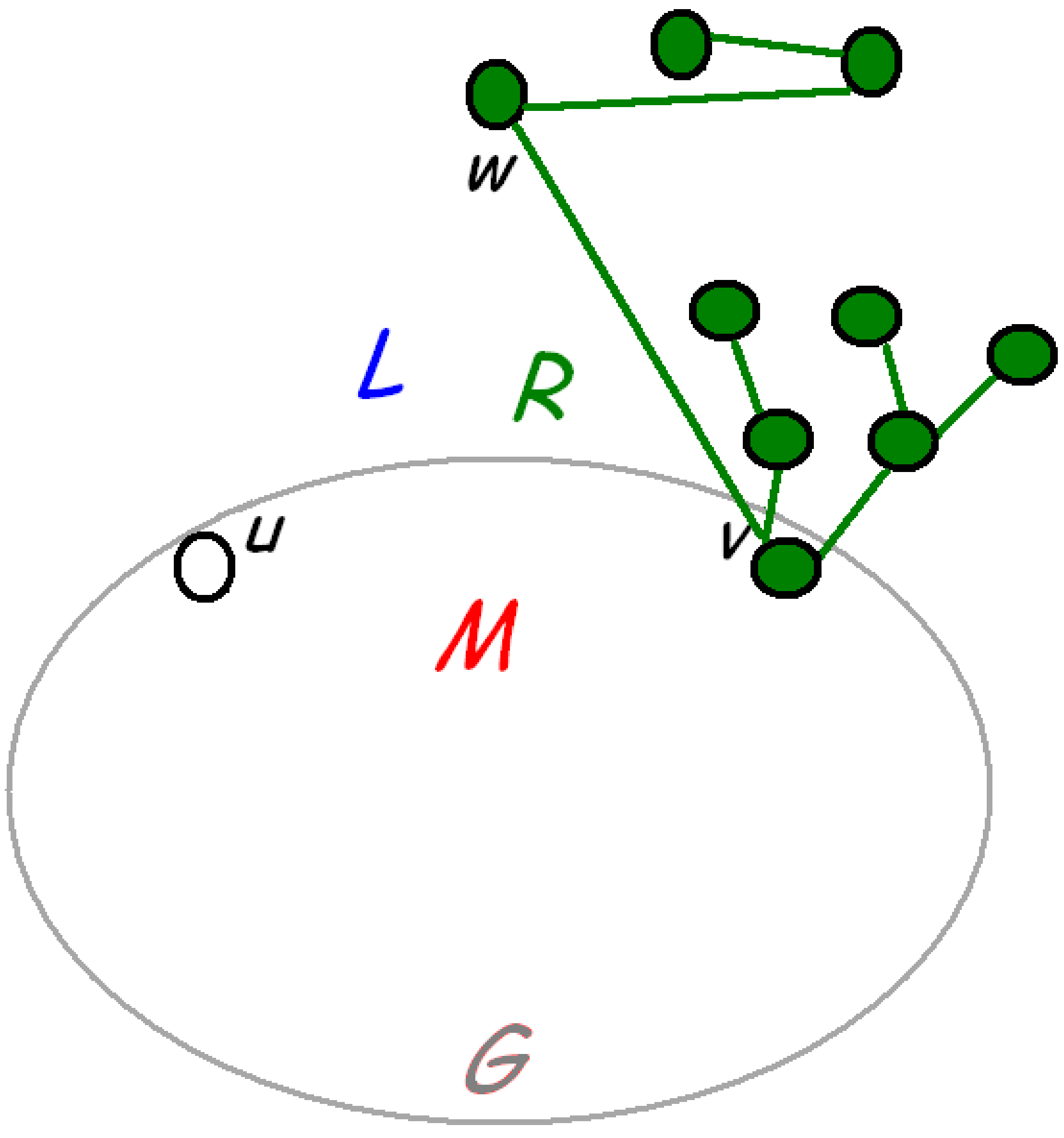
sc(L):



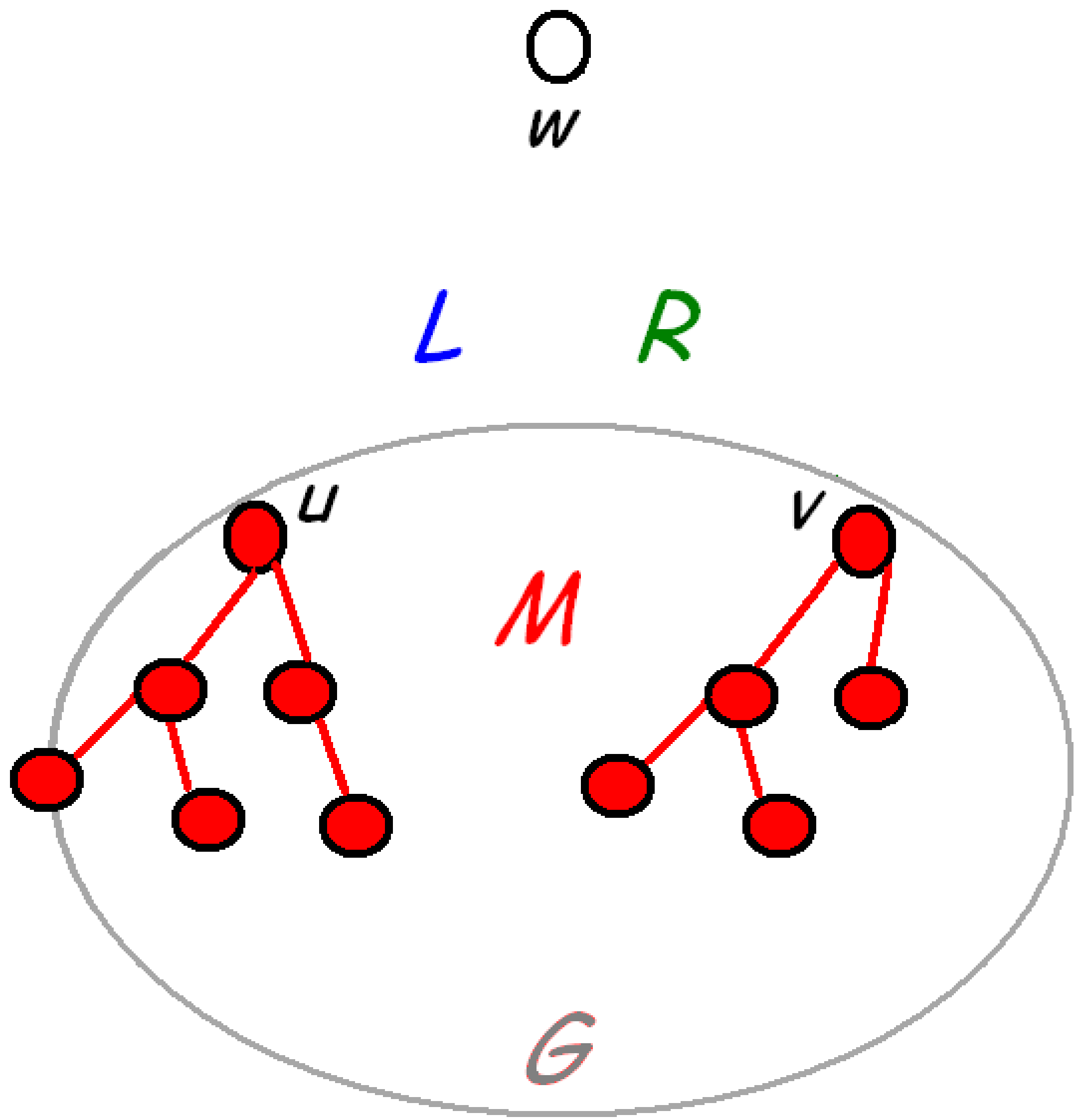
dc(R):



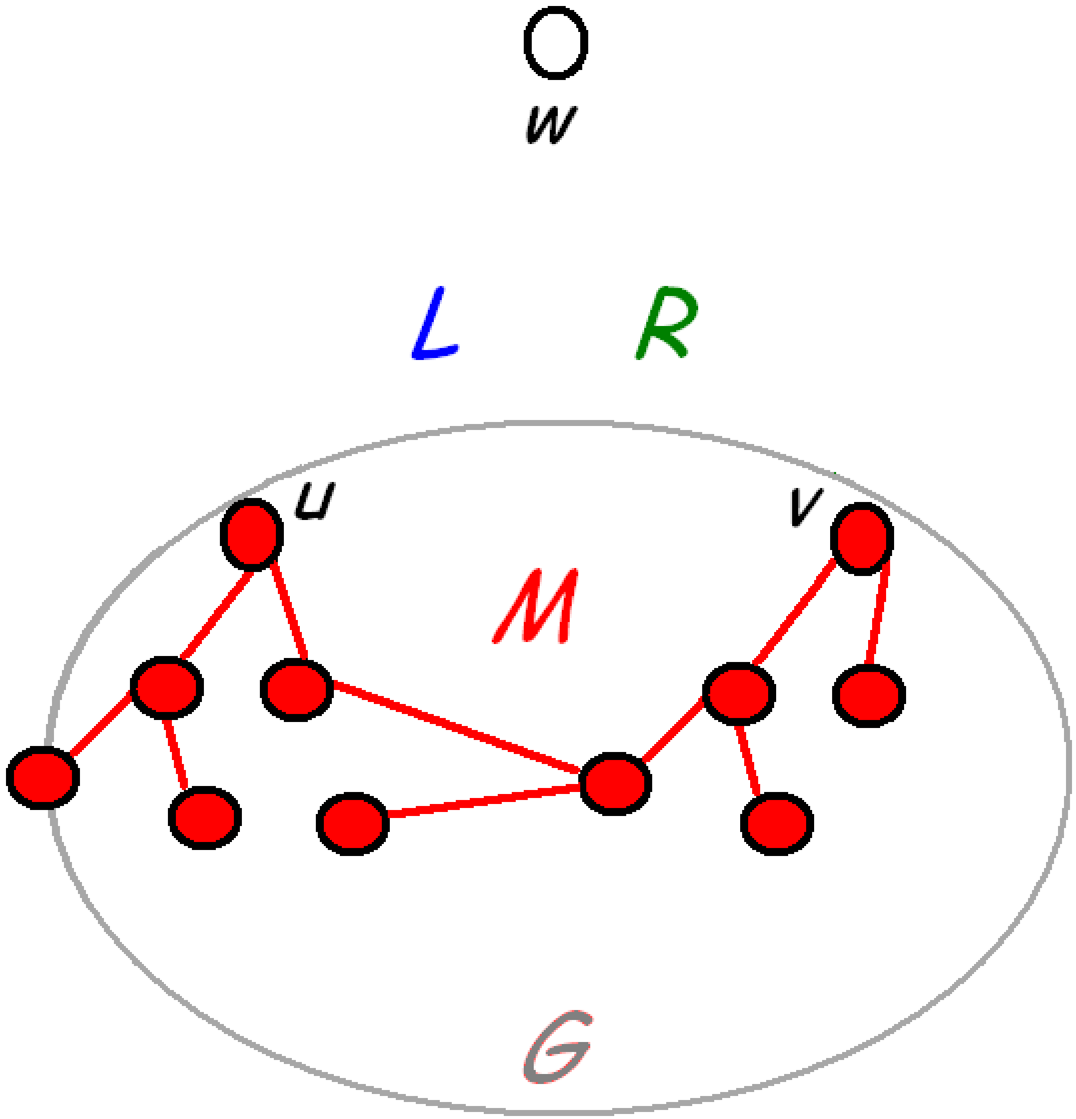
sc(R):



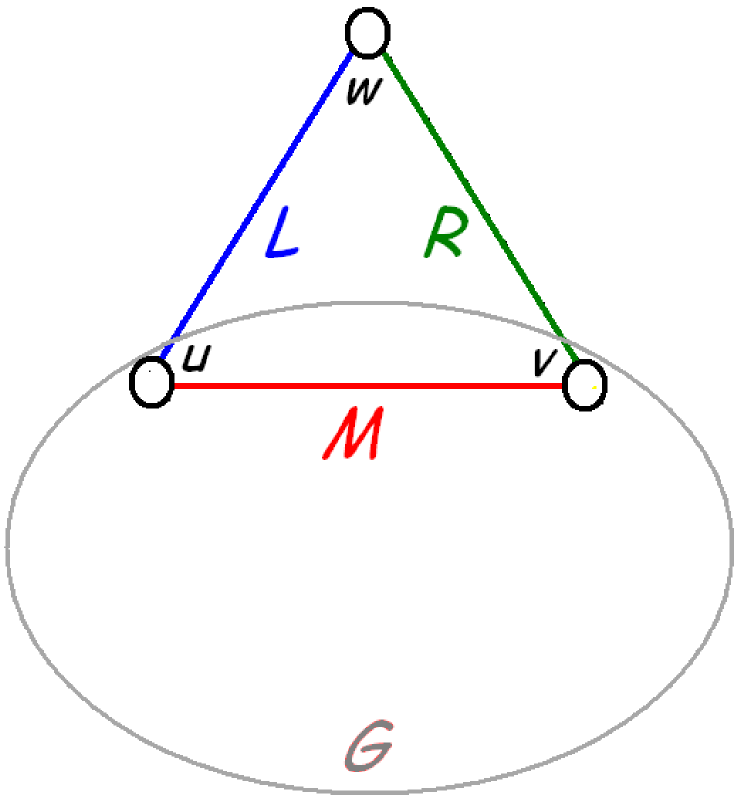
dc(M):



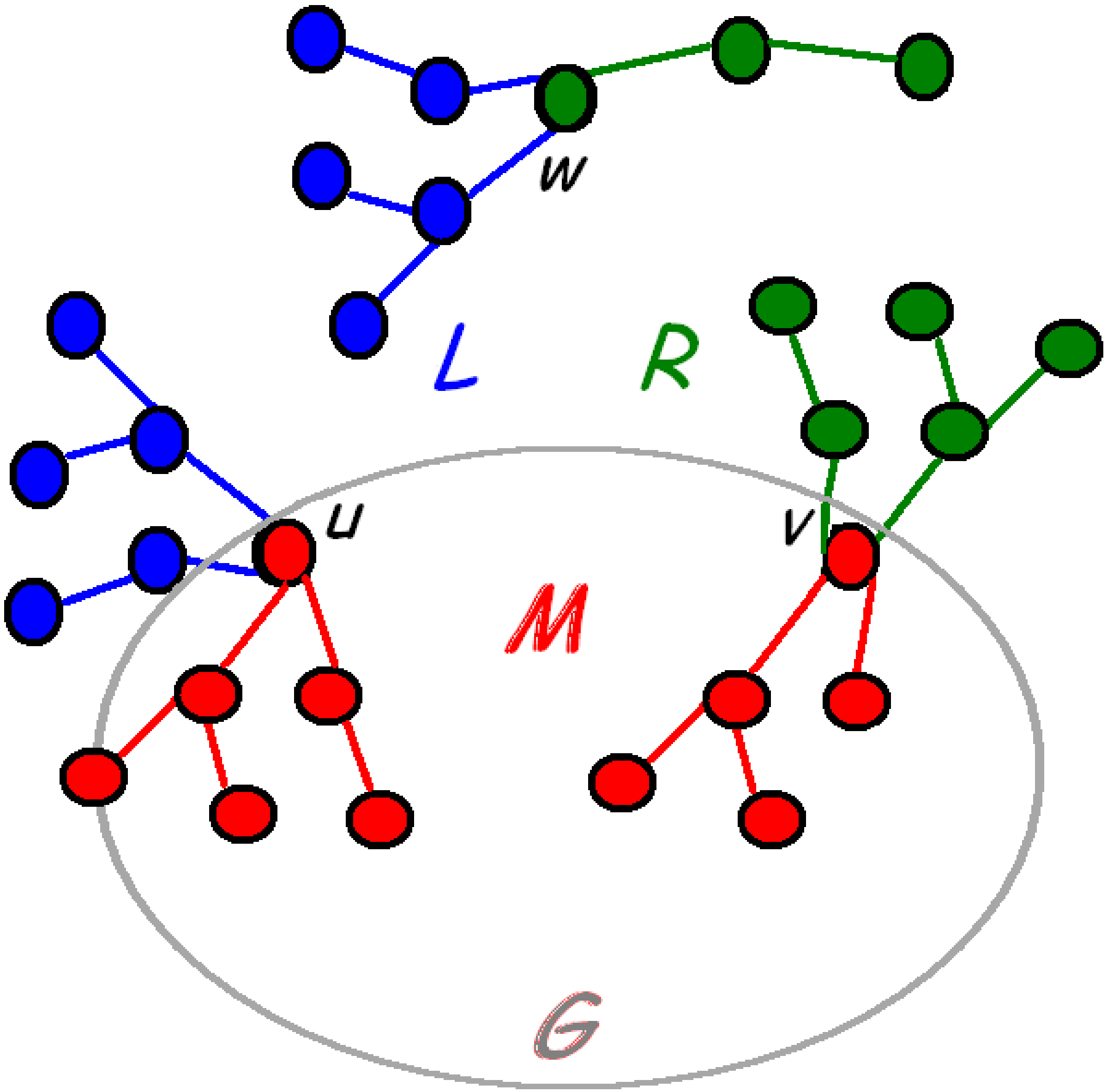
sc(M):



Now let us consider all 8 choices for choosing a contribution from each of the edges L, R, and M.

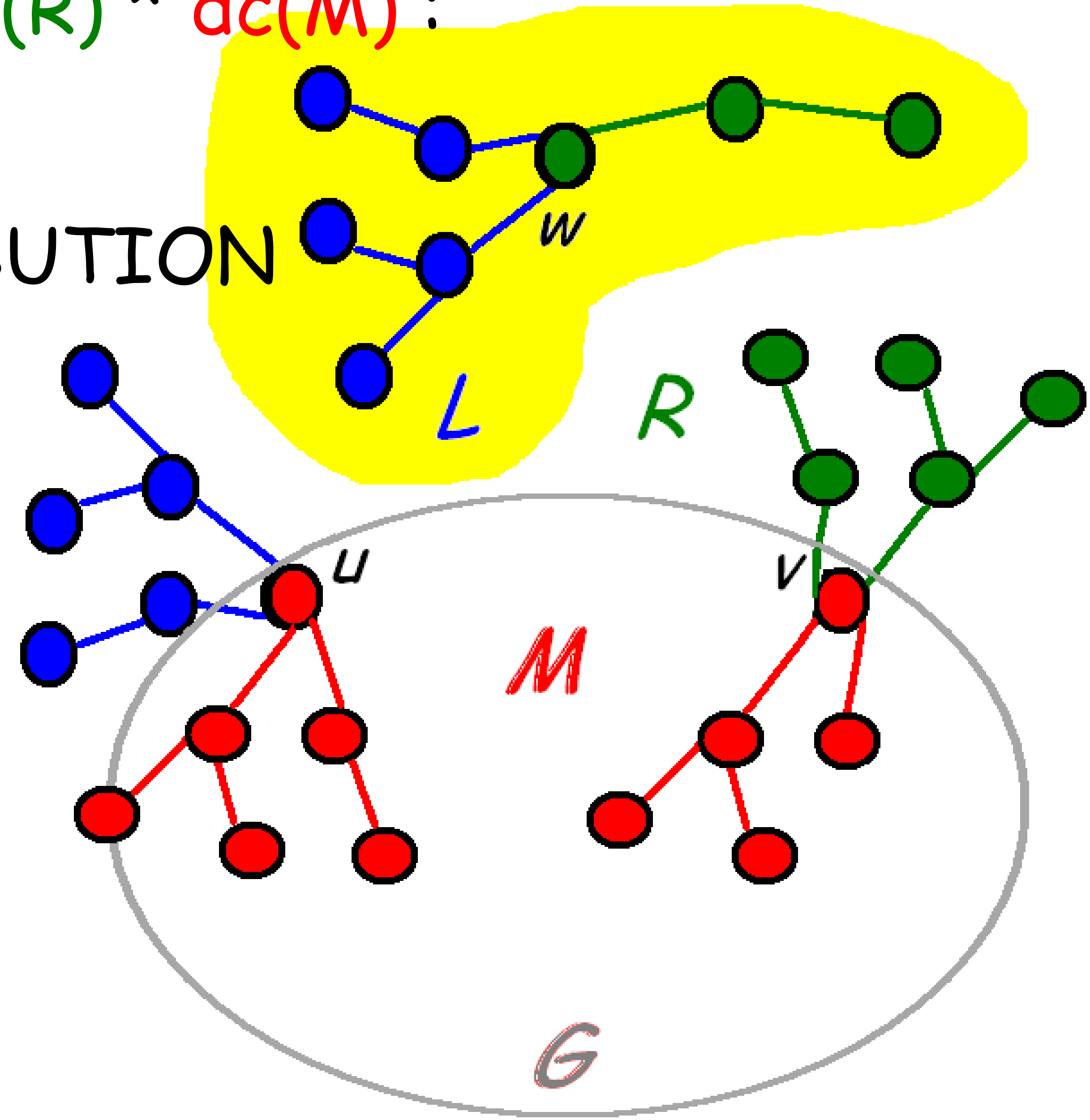


dc(L) * dc(R) * dc(M) :

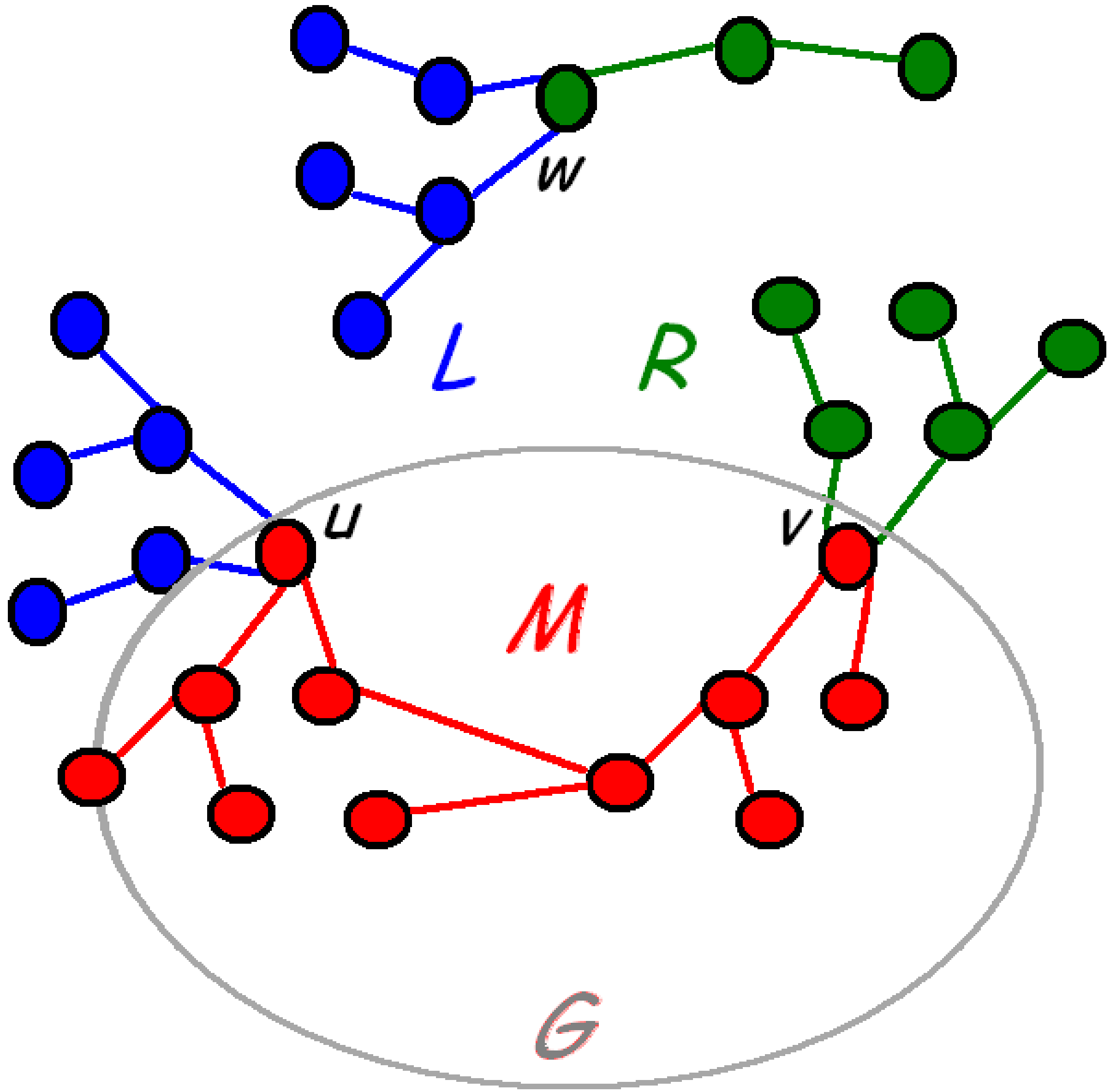


dc(L) * dc(R) * dc(M) :

NO CONTRIBUTION

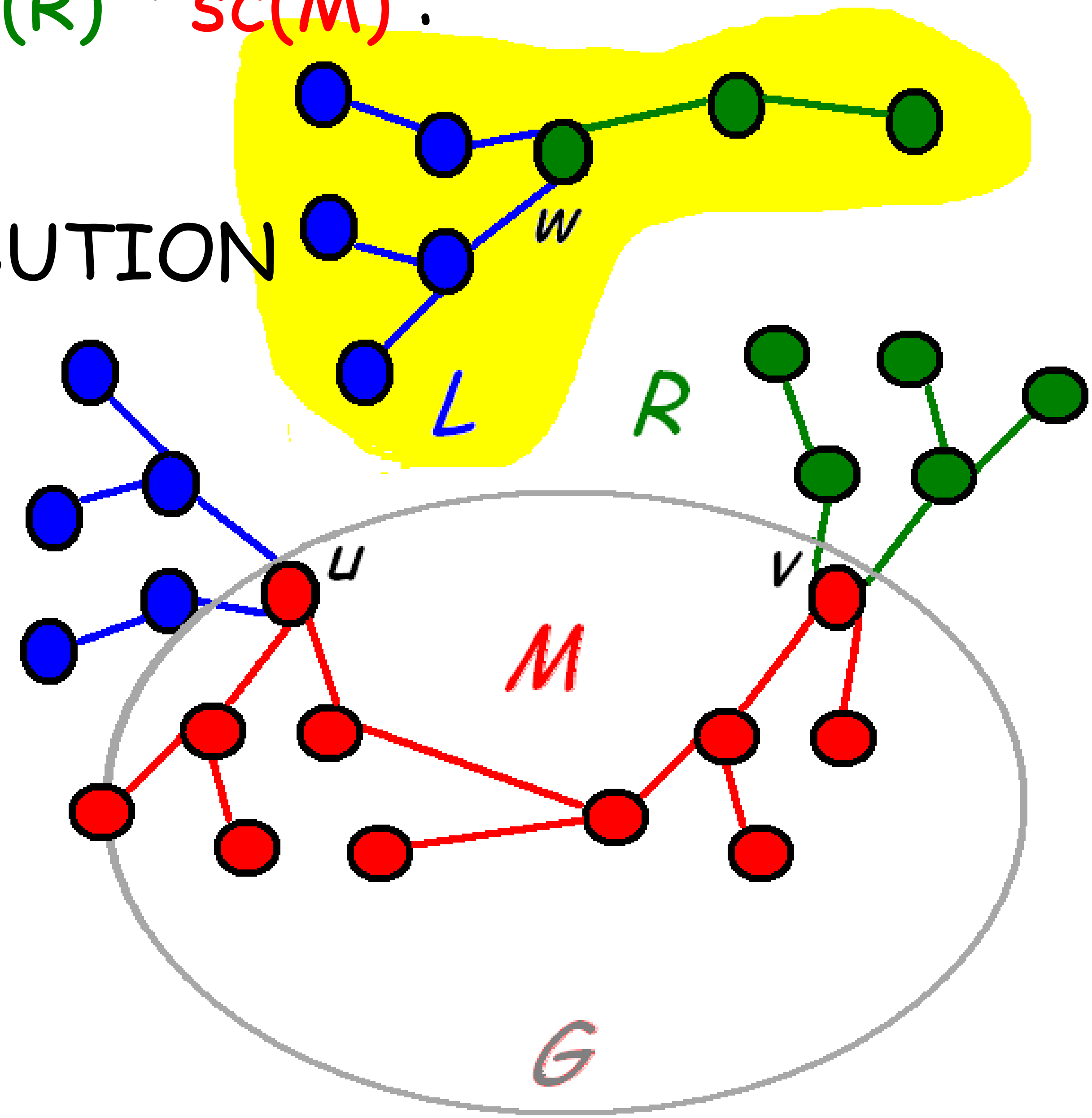


dc(L) * dc(R) * sc(M) :

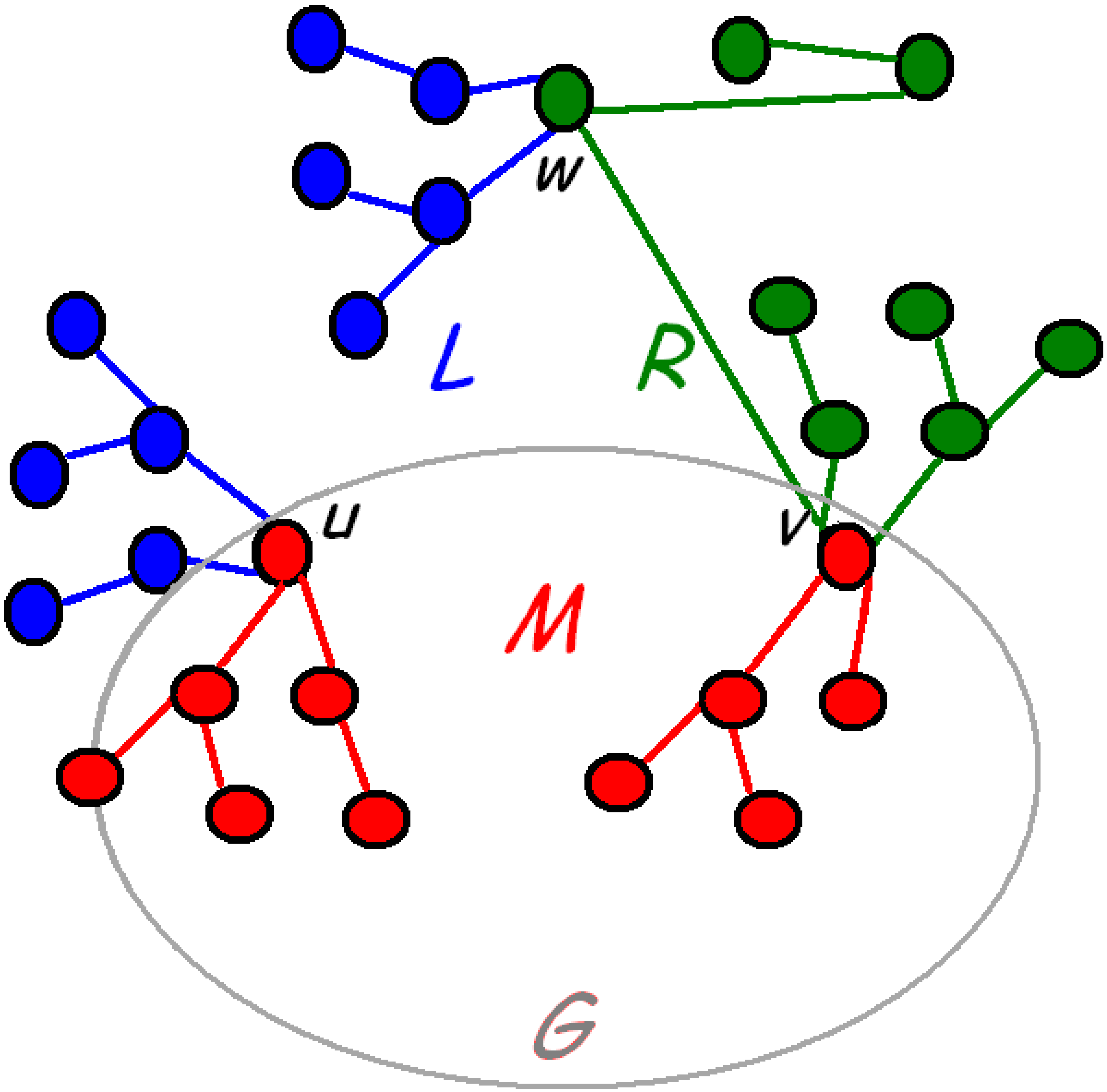


dc(L) * dc(R) * sc(M) :

NO CONTRIBUTION



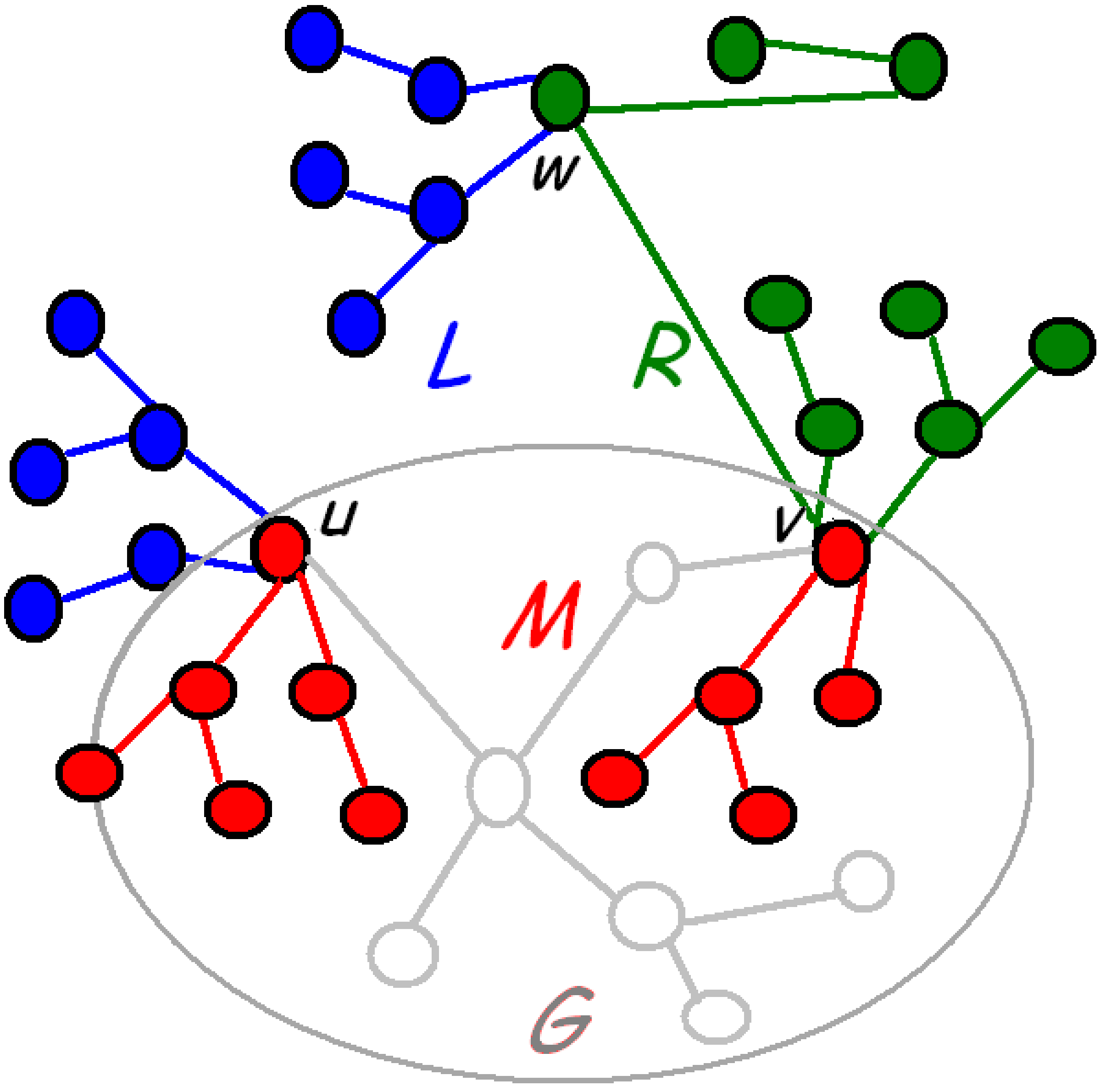
dc(L) * sc(R) * dc(M) :



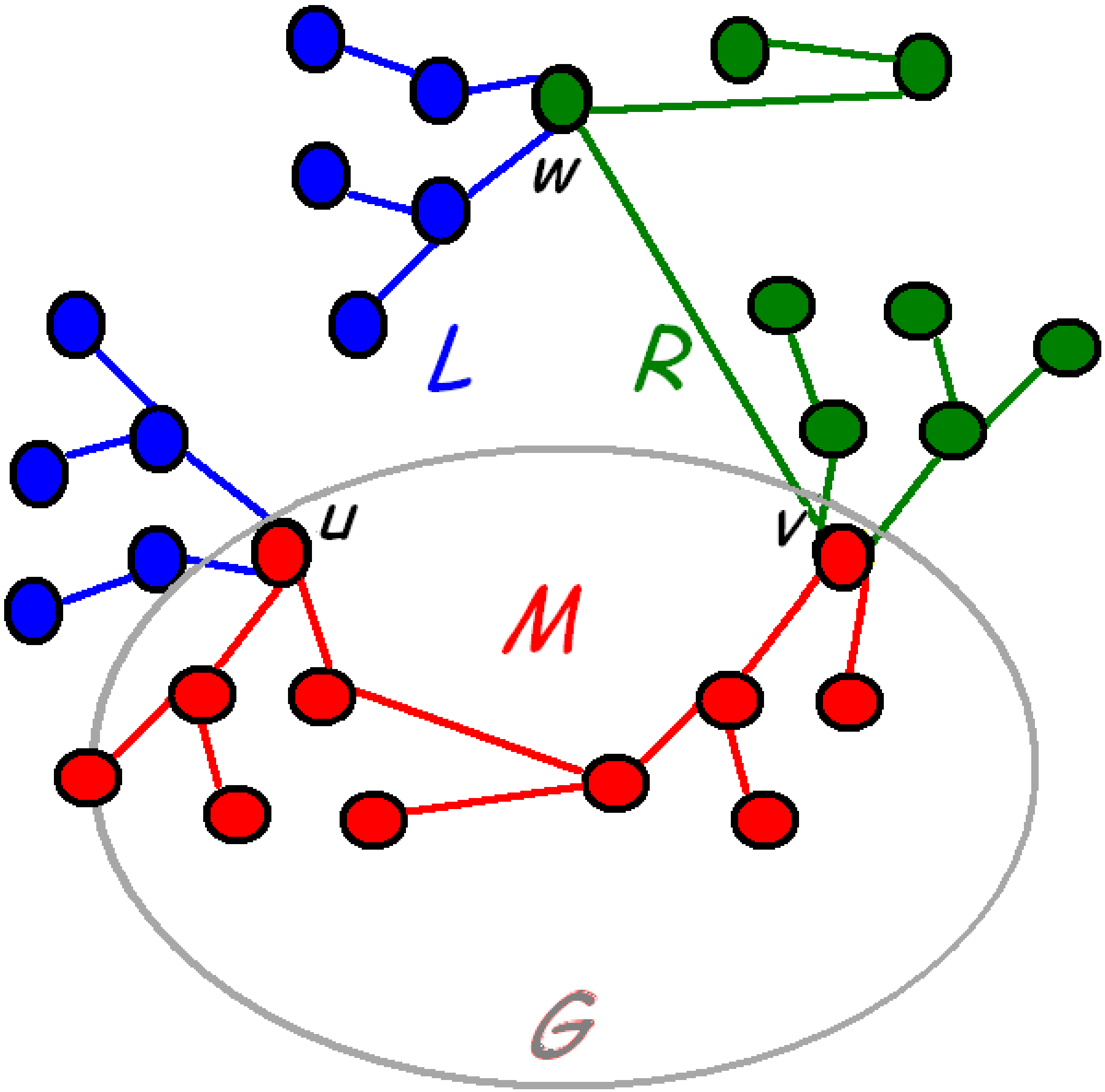
dc(L) * sc(R) * dc(M) :

Add to

dc(M’)



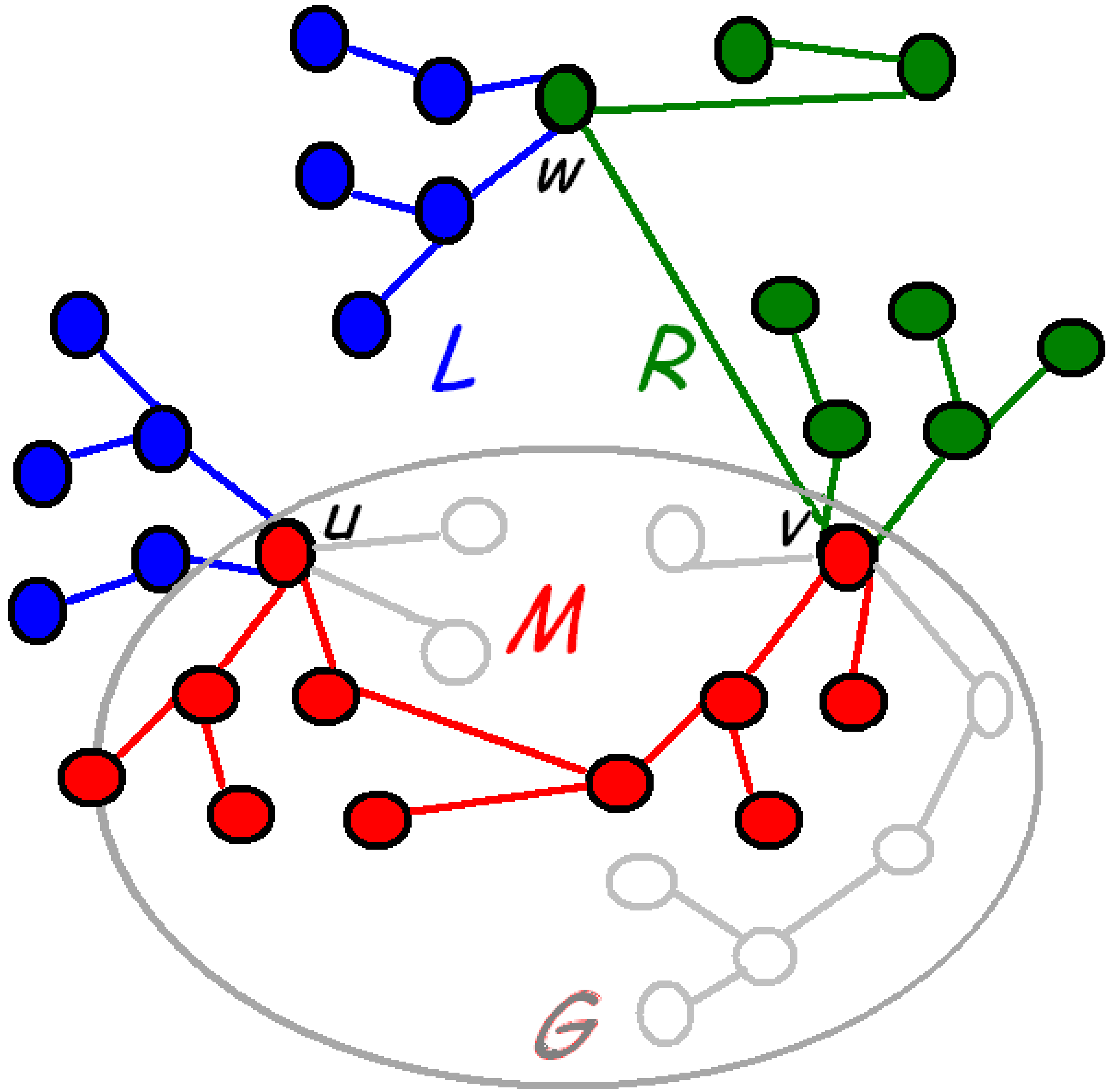
dc(L) * sc(R) * sc(M) :



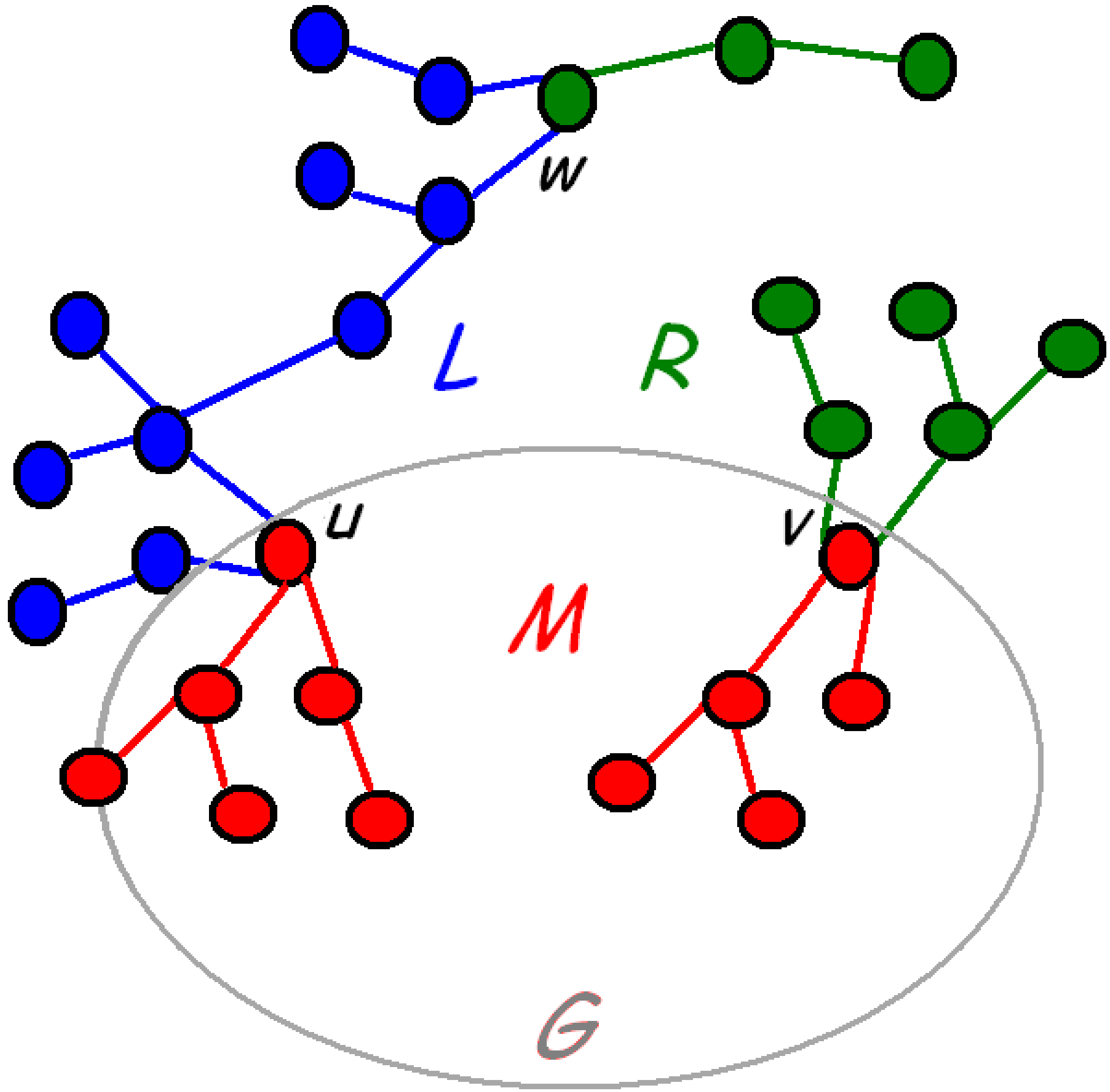
dc(L) * sc(R) * sc(M) :

Add to

sc(M')



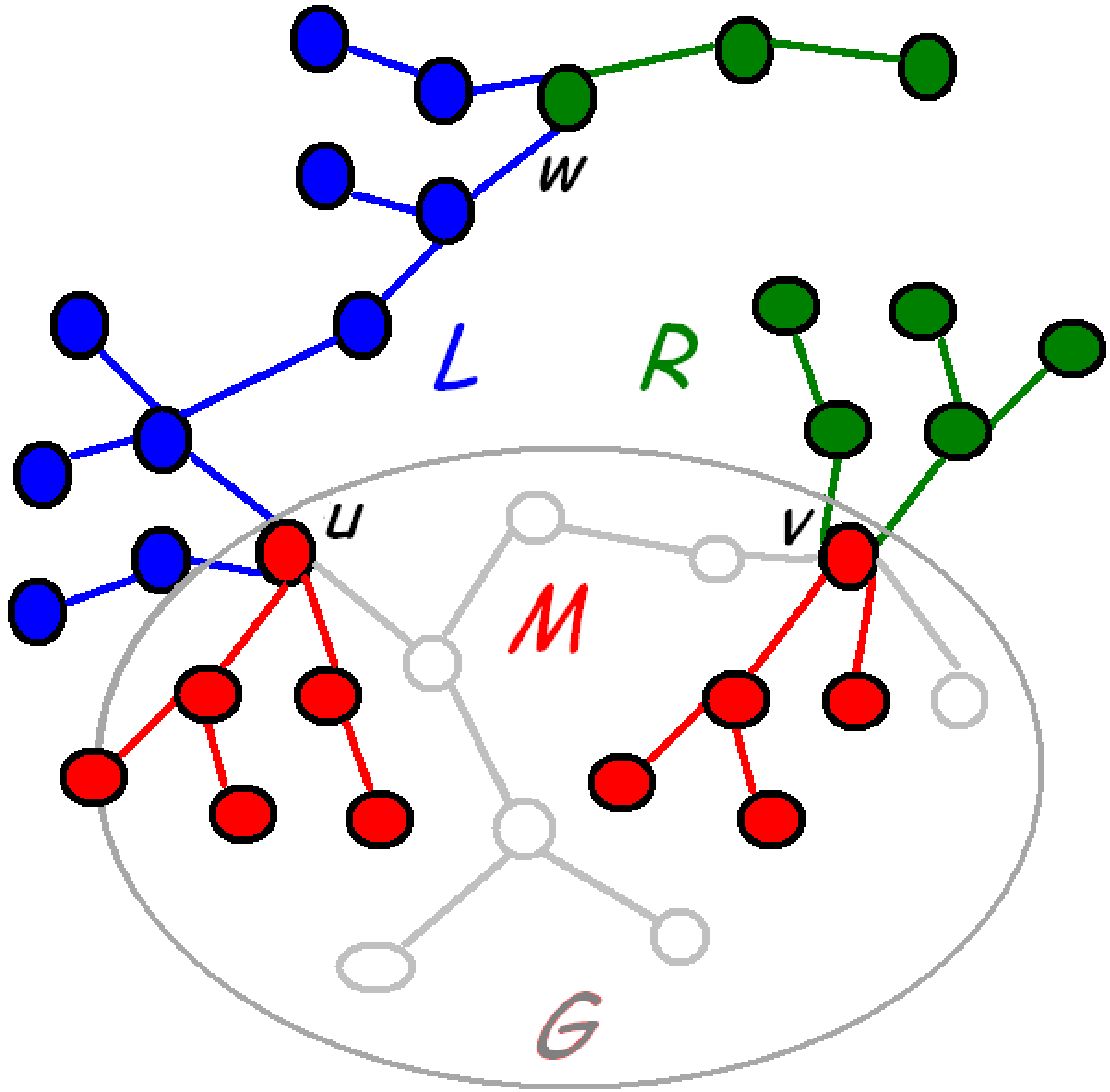
sc(L) * dc(R) * dc(M) :



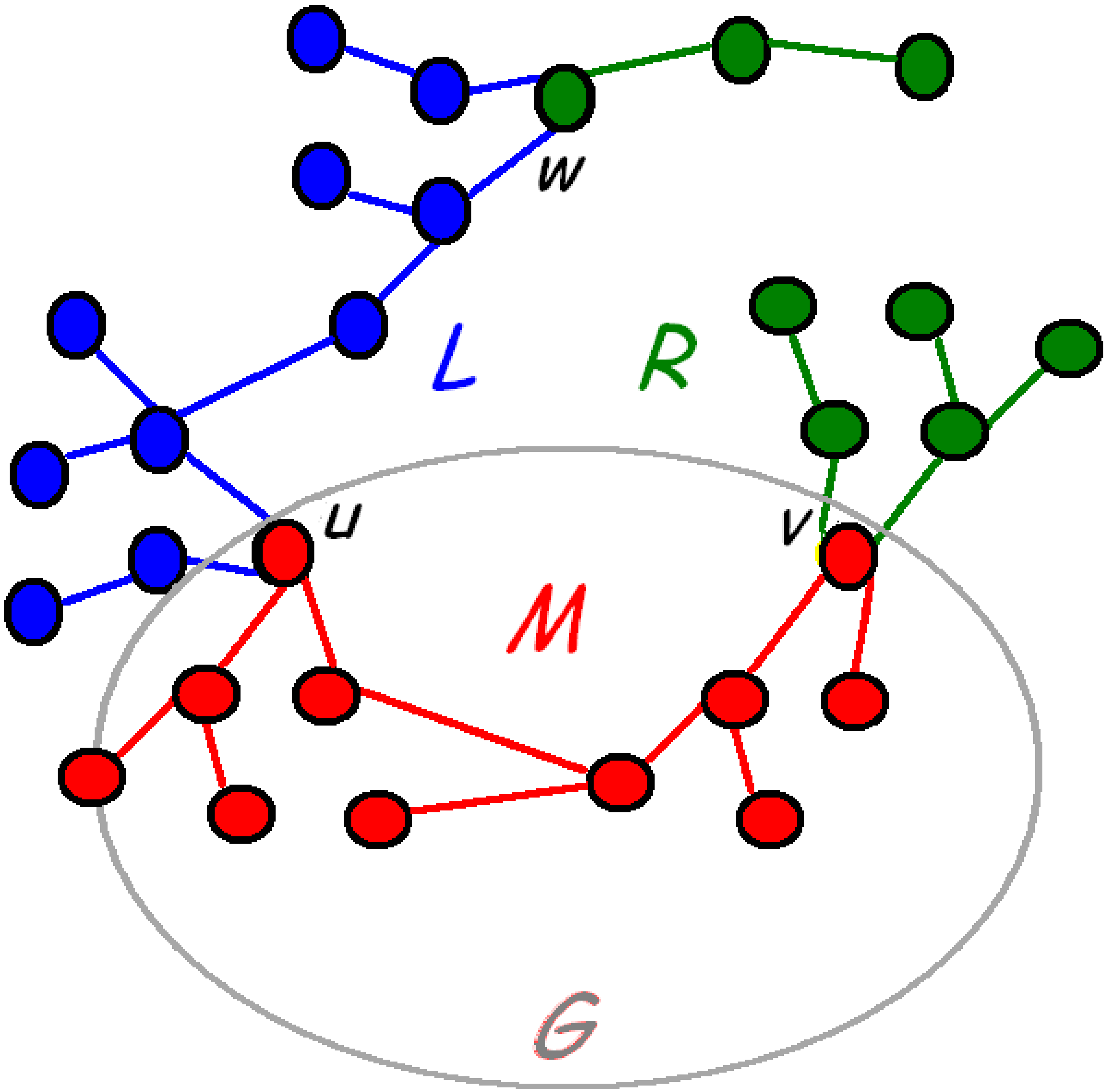
sc(L) * dc(R) * dc(M) :

Add to

dc(M')



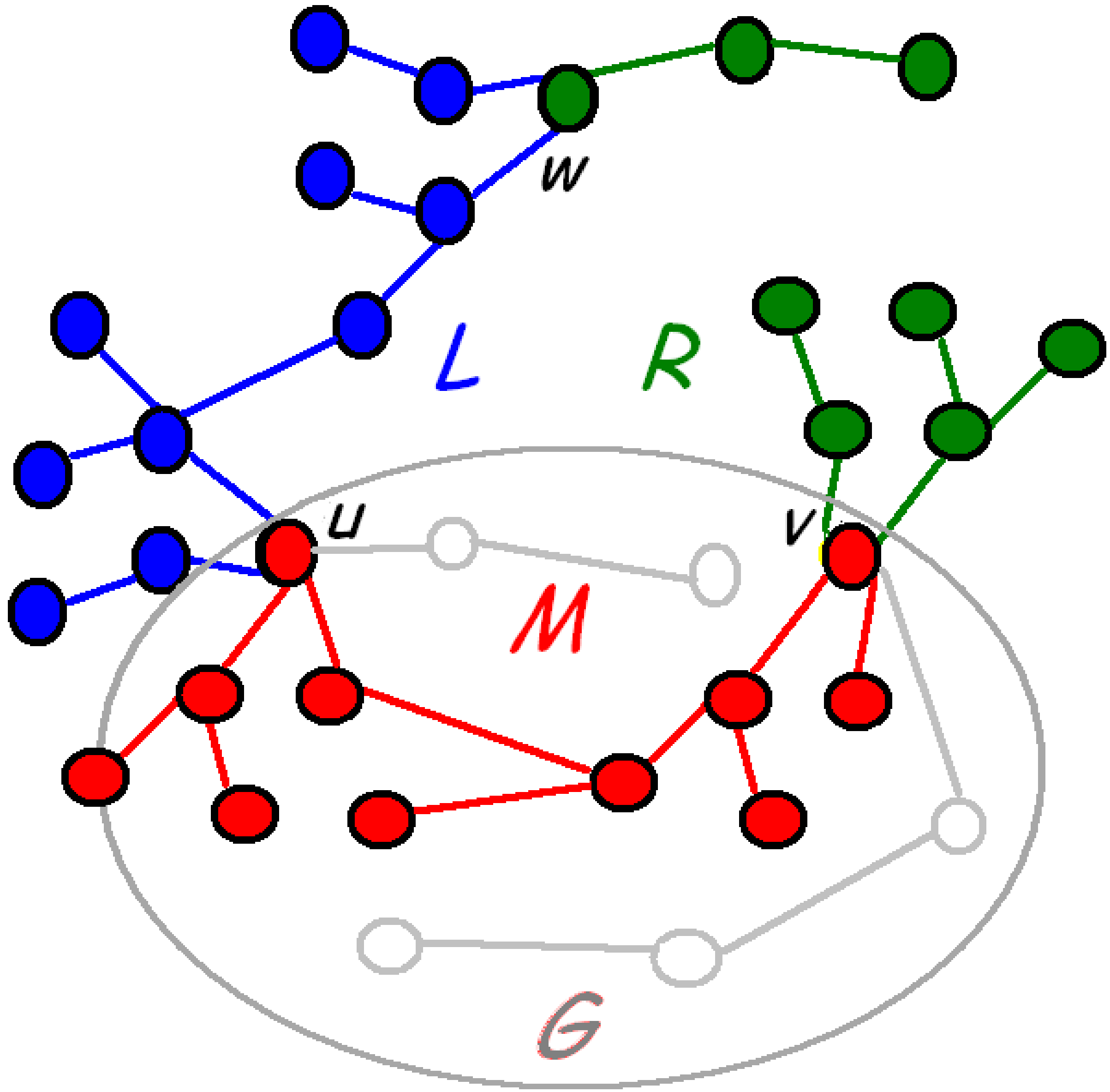
sc(L) * dc(R) * sc(M) :



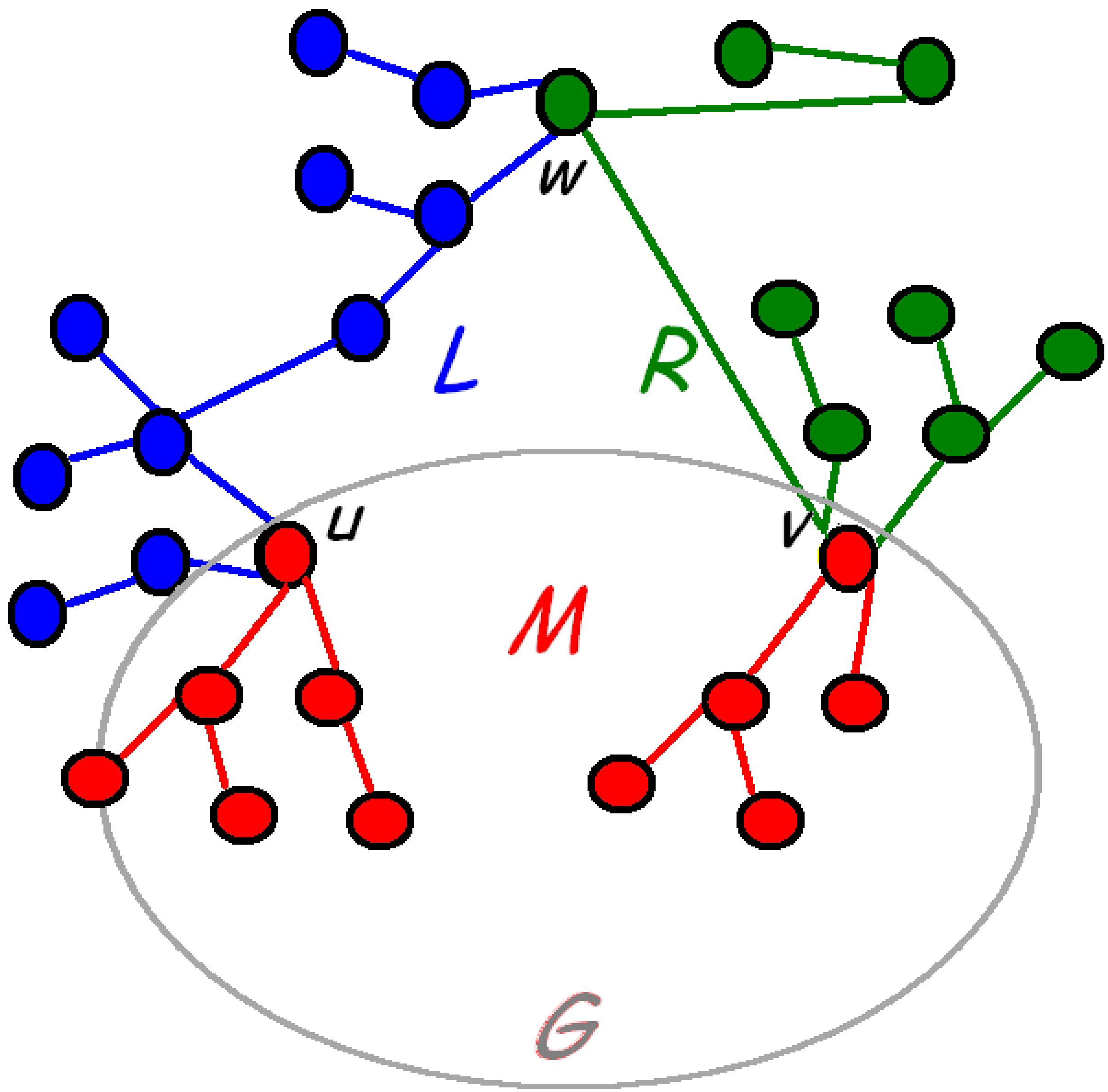
sc(L) * dc(R) * sc(M) :

Add to

sc(M')



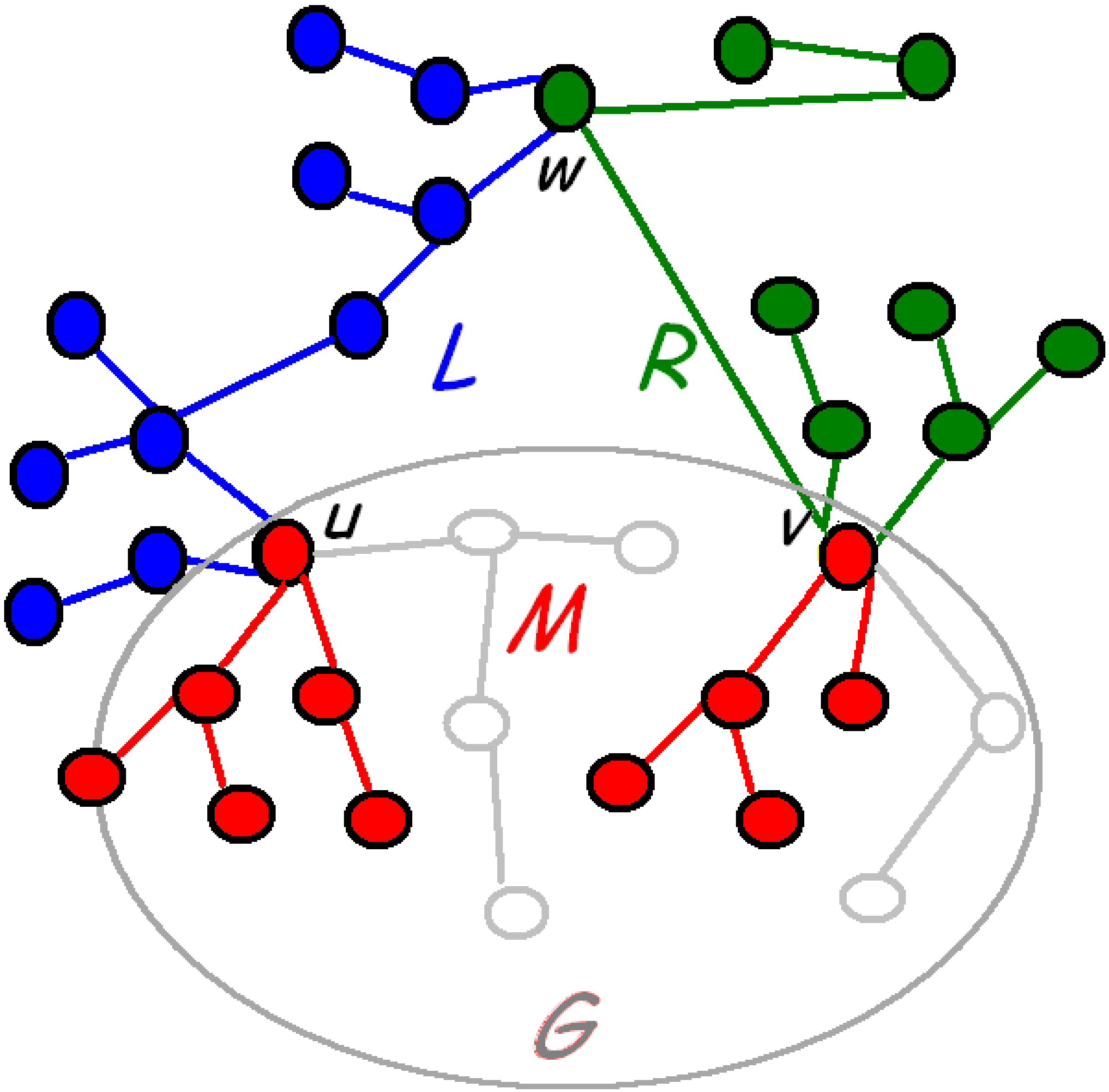
sc(L) * sc(R) * dc(M) :



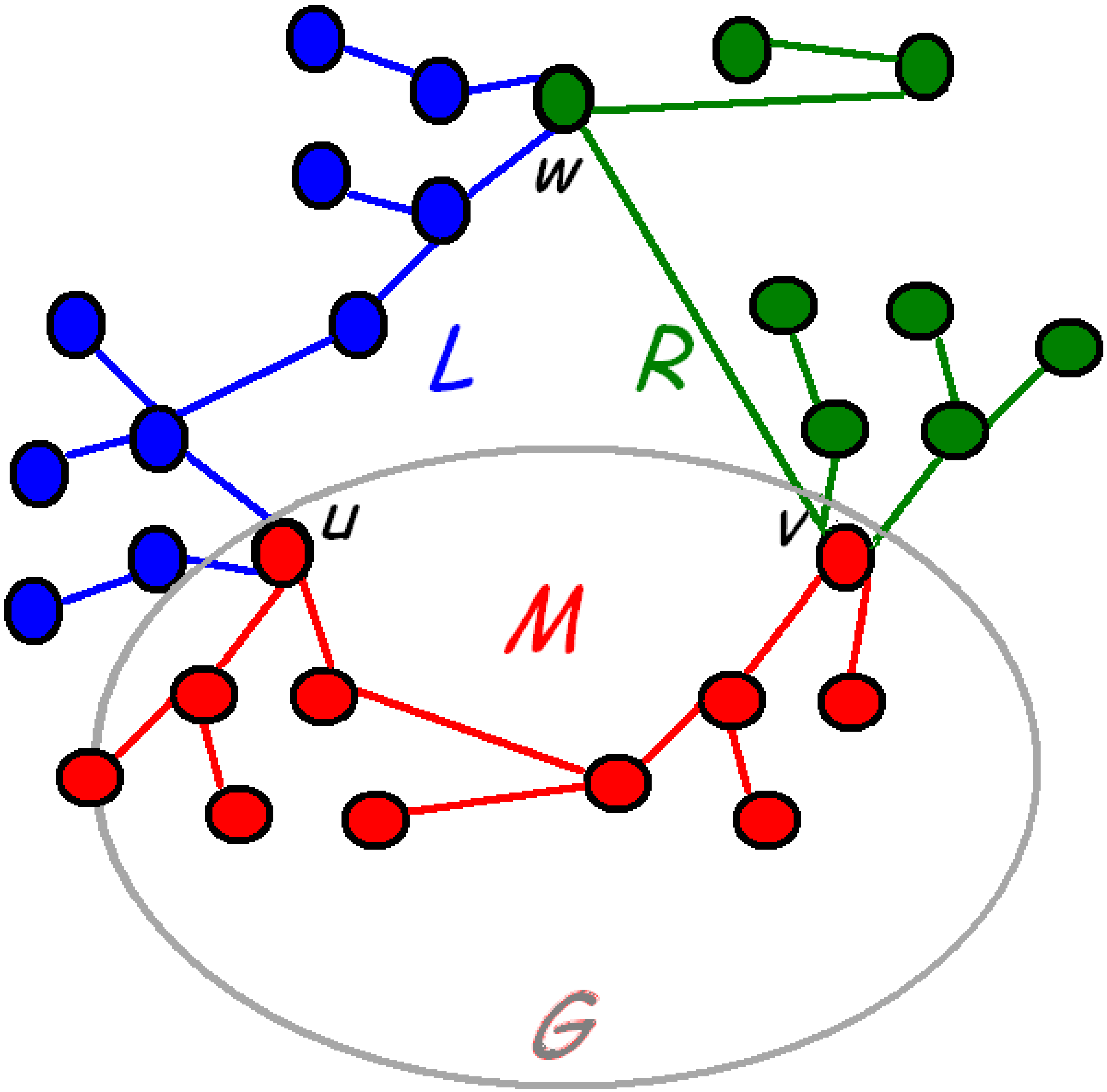
sc(L) * sc(R) * dc(M) :

Add to

sc(M')

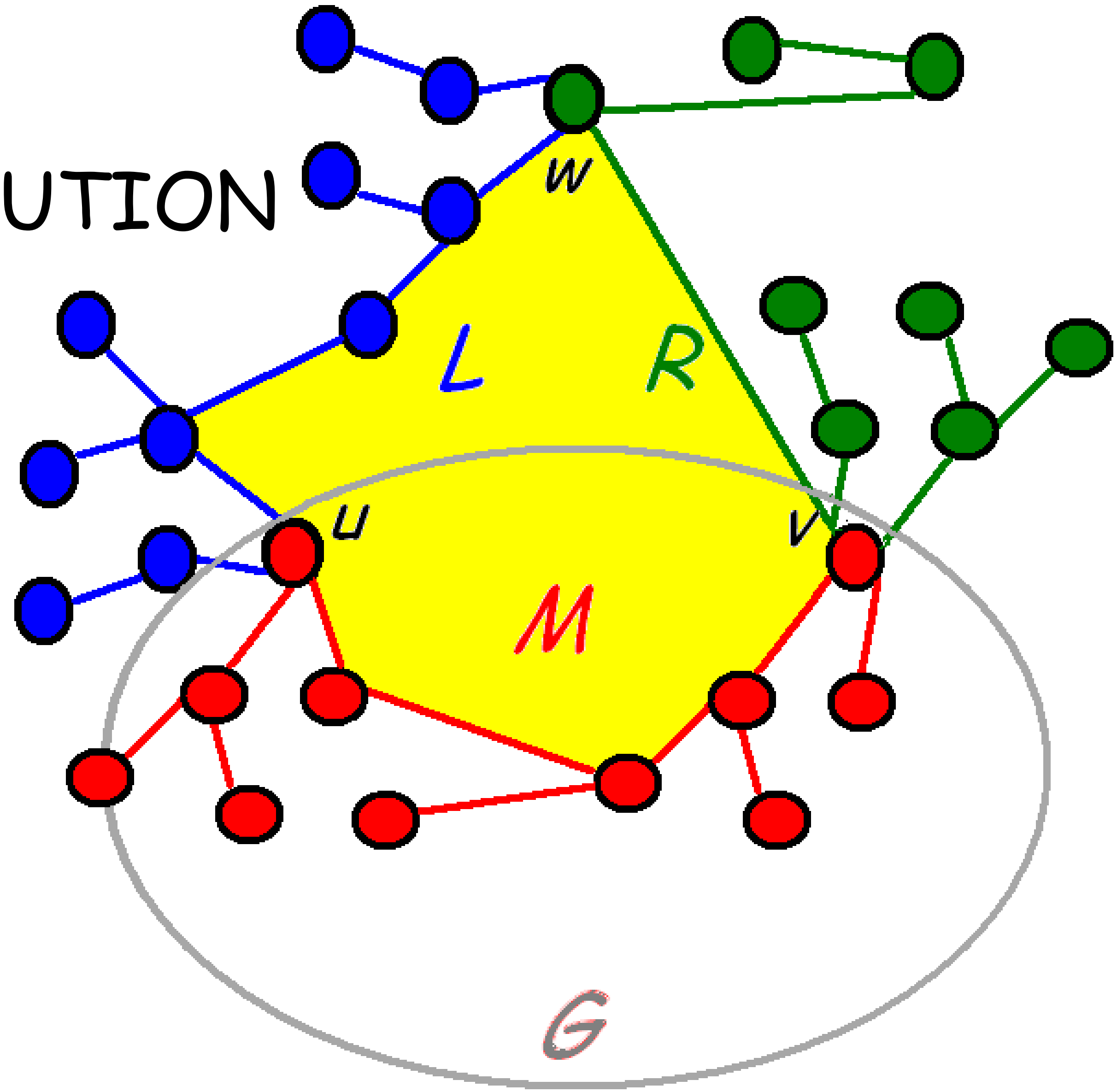


sc(L) * sc(R) * sc(M) :



sc(L) * sc(R) * sc(M) :

NO CONTRIBUTION



No contribution:

dc(L) * dc(R) * dc(M) (w disconnected)

dc(L) * dc(R) * sc(M) (w disconnected)

sc(L) * sc(R) * sc(M) (has a cycle)

dc(M')= dc(L) * sc(R) * dc(M) +

sc(L) * dc(R) * dc(M)

sc(M')= dc(L) * sc(R) * sc(M) +

sc(L) * dc(R) * sc(M) +

sc(L) * sc(R) * dc(M)

Update Formulas