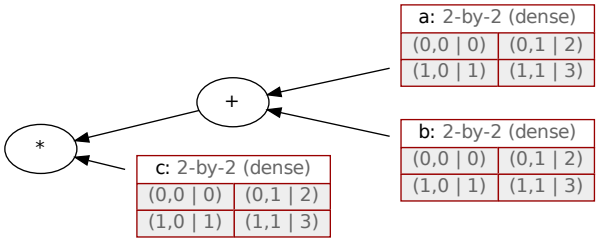
```
#
#
#
#
#
#
#
from casadi import *
from casadi.tools import *
```

Let's revisit briefly the difference between SX and MX

```
a = MX.sym("a",2,2)
b = MX.sym("b",2,2)
c = MX.sym("c",2,2)

d = a+b
e = d*c
```

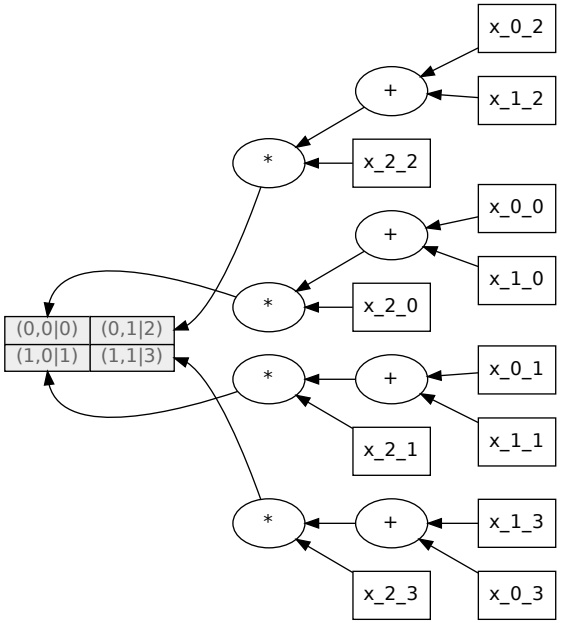
The element-wise addition and multiplication operators appear just as a single node in the MX expression graph

```
dotdraw(e)
```



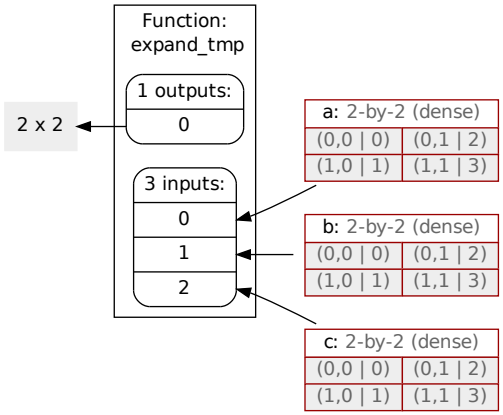
We can use expand to expand into subexpressions

```

f = Function("f", [a,b,c],[e])

g = f.expand('g')

dotdraw(g(*g.sx_in()))
```



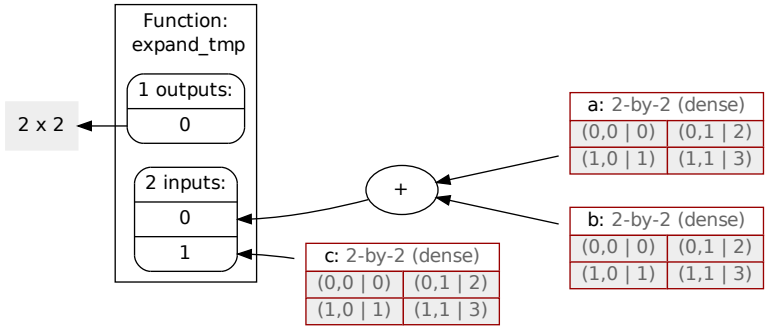
There is also a variant to perform expansion immediately on the MX graph The expanded SX graph is hidden inside an SX graph call

```
dotdraw(matrix_expand(e))
```



An additional features of this variant is that one can choose which expressions remin outside of the expansion scope. In the following we list 'a+b=d' as a node on the boundary of expansion:

```
37 dotdraw(matrix_expand(e,[d]))
```



Note how the additions is not expanded, while the multiplication ended up in the expression