Write a program to compute the Multivariable Alexander Polynomial (MVA) using R matrix, as a straightforward generalization of the Alexander program. To avoid confusion with the Alexander program, we put the letter V (stands for (multi)-Variable) in the name of all the functions that follows.

First we input the positive R matrix, where c1 is the label of the overstrand and c2 is the label of the understrand

```
RpVM[c1_, c2_] := {{tc1^-1/4 tc2^-1/4, 0, 0, 0}, {0, 0, tc1^-1/4 tc2^1/4, 0},
  {0, tc1^1/4 tc2^-1/4, (tc1^-1/4 - tc1^3/4) tc2^-1/4, 0}, {0, 0, 0, -tc1^1/4 tc2^1/4}}
```

Now the rule to call the entries

```
RpV[i_, j_, k_, l_, c1_, c2_] :=
 Which[i == 1 && k == 1, RpVM[c1, c2][[i + j - 1, k + l - 1]], i == 1, RpVM[c1, c2][[i + j - 1, k + l]],
  k == 1, RpVM[c1, c2][[i + j, k + l - 1]], True, RpVM[c1, c2][[i + j, k + l]]]
```

Next we input the negative R matrix, where c1 is the overstrand and c2 is the understrand

```
RmVM[c1_, c2_] := {{tc1^1/4 tc2^1/4, 0, 0, 0},
  {0, (-1 + tc1)/(tc1^1/4 tc2^1/4), tc2^1/4/tc1^1/4, 0}, {0, tc1^1/4/tc2^1/4, 0, 0}, {0, 0, 0, -1/(tc1^1/4 tc2^1/4)}}
```

and the rule to call the entries

```
RmV[i_, j_, k_, l_, c1_, c2_] :=
 Which[i == 1 && k == 1, RmVM[c1, c2][[i + j - 1, k + l - 1]], i == 1, RmVM[c1, c2][[i + j - 1, k + l]],
  k == 1, RmVM[c1, c2][[i + j, k + l - 1]], True, RmVM[c1, c2][[i + j, k + l]]]
```

Next we input the h matrix, where c is the label of the strand

```
hV[c_, c_] := {{tc^1/2, 0}, {0, -tc^1/2}}
```

Now we are ready to write the main program

```
MVA[Link_] := Module[{m},
  m = Max[Drop[#, -2] & /@ (List @@@ List @@ Link)];
  Sum[Link /. {
         Xp[i_, j_, k_, l_, c1_, c2_] :> RpV[si, sj, sk, sl, c1, c2],
         Xm[i_, j_, k_, l_, c3_, c4_] :> RmV[si, sj, sk, sl, c3, c4],
         Cu[k_, c_, c_] :> hV[c, c][[sk, sk]]^-1,
         Ca[k_, c_, c_] :> hV[c, c][[sk, sk]]
        }, ##] & @@ ({s#, 1, 2} & /@ Range[m]) / (2 t1^1/2 - 2 t1^-1/2) // Simplify
 ]
```

Now let's compute the MVA of a few links. Notice there is a change in how we enter the link. Analogous to the Alexander polynomial, we need to keep one component open. Here our convention is that we always keep the component labeled 1 open. We then label the crossings, cups, caps and read our link from top to bottom as usual.

The Hopf Link

```
MVA[Ca[3, 2, 2] Xp[1, 3, 4, 2, 2, 1] Xp[4, 2, 1, 3, 1, 2]]
```

```
-1
```

The Borromean Ring (very slow :(, the algorithm uses the state-sum method, so we have to sum over approximately four billion terms, it will probably be faster if we just take the product of R matrices and then compute the partial trace)

```
Timing[MVA[Ca[5, 2, 2] Ca[13, 3, 3] Xp[1, 5, 12, 4, 2, 1] Xp[12, 4, 3, 11, 1, 2]
   Xp[11, 13, 16, 10, 3, 2] Xp[16, 10, 9, 15, 2, 3] Xm[3, 9, 8, 2, 1, 2]
   Xm[8, 2, 1, 7, 2, 1] Xm[7, 15, 14, 6, 2, 3] Xm[14, 6, 5, 13, 3, 2]]]
```

$$\left\{47.0313,\ \frac{(-1+t_1)\ (-1+t_2)\ (-1+t_3)}{\sqrt{t_1}\ \sqrt{t_2}\ \sqrt{t_3}}\right\}$$