ETVO Calculator (and its online version) is a basic interpreter to handle series and matrices in dioids MinMax[[g,d]], E[[d]] and T[[g]]. It is a program which executes some commands explained below.

IMPORTANT : the results are mostly compatible with the grammar of the parsers, so that it is possible to save intermediate results in text files for future use.

Variables : the type is defined by the first letter of the identifier (composed only by alpha numerical symbols)

s__ : series in MinMax[[g,d]] S__ : matrix with entries in MinMax[[g,d]]

e__ : series in E[[d]] E__ :matrix with entries in E[[d]]

t__ : series in T[[g]] T__ :matrix with entries in T[[g]]

```
->s1=g2.[g2.d3]*
s1=(g2.d0).[g2.d3]*
->eA=m2.b3.[g1.d2]*
eA=(m2.b3.d0).[g1.d2]*
->tx=d4.D2.d1.[g1.d2]*
tx=(d6.v2.w2.d-1.g0).[d2.g1]*
```

Expressions : *a command starting with % is ignored*

g2, d1 = event-shift and time-shift operators in MinMax[[g,d]]
g2, d1, m2, b3, N4 = m2 (event-multiplier), b3 (event-divider), N4=m4.b4 are the basic operators in E[[d]]
g2, d1, v2, w3, D4 = v2 (time-multiplier), w3(time-divider), D4=v4.w4 are the basic operators in T[[g]]
m<1,2>, b<3,1,2> = cyclic multiplier/batch in E[[d]]
d<3,2> = cyclic delay in T[[g]]
exp1.exp2 = product (which is non commutative for E[[d]] and T[[g]])
exp1 + exp2 = sum
inf(exp1,exp2) = inf
[exp]* = Kleene star
frac(exp1,exp2) = residuation exp1/exp2 in MinMax[[g,d]]
lfrac(exp1,exp2) rfrac(exp1,exp2) = residuation in E[[d]] and T[[g]] and matrices
pr+(exp) = projection in causal set
?=(exp1,exp2) checks equality (unavailable for matrices)
EA(0,1) = entry of a matrix
eps(r,c) = epsilon matrix of size r x c

```
->EA=eps(2,2)
->EA(0,0)=g3.m2.b2
->EA(1,1)=g2.d5
->EA(1,0)=d5
->EA(0,1)=m2.b2.g2.d7
->EAs=[EA]*
EAs(0,0)=((g0.d0))+[g2.d12]*.(g2.m2.b2.d12)
EAs(0,1)=[g2.d12]*.(g2.m2.b2.d7)
EAs(1,0)=((g0.d5))+[g2.d12]*.(g2.m2.b2.d17)
EAs(1,1)=((g0.d0))+[g2.d12]*.(g2.m2.b2.d12)
```

Evaluation : to evaluate/print an expression without saving it in a variable

M{exp} = evaluates exp as a MinMax[[g,d]] expression
E{exp} = evaluation in E[[d]]
T{exp} = evaluation in T[[g]]

```
->T{d<1,2>}
(d1.v2.w2.g0)
```

Functions :

EdToMM(expEd) = zero slice of a series E[[d]] (impulse response), returns a series in MinMax[[g,d]]
TgToMM(expTg) = zero slice of a series T[[g]] (impulse response), returns a series in MinMax[[g,d]]
right(exp) left(exp) = returns the series in the right or left form
MMToEd(expMM) = injection of a MinMax[[g,d]] series into E[[d]]
MMToTg(expMM) = injection of a MinMax[[g,d]] series into T[[g]]
asMuVar(expEd) = displays a E[[d]] series with m<seq> coefficients
asDeltaVar(expTg) = displays a T[[g]] series with d<seq> coefficients
asCoreEd(expEd) = displays the Core decomposition of a E[[d]] series
asCoreTg(expTg) = displays the Core decomposition of a T[[g]] series
randMM(nbTerms) = randomly generated MinMax[[g,d]] series
randEd(m,b,nbTerms) = randomly generated E[[d]] series gain m/b
randTg(v,w,nbTerms) = randomly generated T[[g]] series gain v/w
[console version only]
MakePovEd(varEd,xmax,ymax,zmax) : creates a POV-Ray file (e___.pov)
MakePovTg(varTg,xmax,ymax,zmax) : creates a POV-Ray file (t___.pov)

```
->e1=m2.b3.[g1.d2]*
e1=(m2.b3.d0).[g1.d2]*
->e1=left(e1)
e1=[g2.d6]*.(m2.b3.d0+m2.b3.g1.d2+m2.b3.g2.d4)
->s1=EdToMM(e1)
(g0.d4).[g2.d6]*
->asMuVar(e1)
[g2.d6]*.(g0.m<0,0,2>.d0+g0.m<0,2,0>.d2+g0.m<2,0,0>.d4)
-> er=randEd(4,5,2)
er=(((m4.b5.g3+g1.m4.b5).d0))+[g3.d5]*.(g2.m4.b5.d4+(g4.m4.b5.g2+g5.m4.b5).d5)
->MakePovEd(e1,20,20,20)
```

Script : a sequence of commands can be stored in a txt file. Then copy-paste it into the calculator.

```
% Example from Modeling and Control of Weight-Balanced TEGs (IEEE TAC 2014)
% A matrix (4x4) B matrix (4x1) and C matrix (1x4)
EA=eps(4, 4)
EB=eps(4, 1)
EC=eps(1, 4)
EB(0,0)=g0
EC(0,3)=g0
EA(1,0)=b2.d2
EA(1,1)=g1.d2
EA(2,0)=m3
EA(2,2)=g2.d1
EA(3,1)=m3
EA(3,2)=b2.g1.d1
% transfer computation
EH=EC.[EA]*.EB
% neutral feedback computation
EF=pr+(rfrac(lfrac(EH,EH),EH))
```

Output (feedback control)

```
->EF=pr+(rfrac(lfrac(EH,EH),EH))
EF(0,0)=(((g3.m2.b3.g1+g4.m2.b3).d0+g4.m2.b3.d2))+(g6.m2.b3.d4).[g3.d3]*
->asMuVar(EF(0,0))
((g3.m<0,1,1>.d0+g4.m<0,0,2>.d2))+(g6.m<0,0,2>.d4).[g3.d3]*
```