Graphs representation

```
$ cat talk.dot
digraph talks {
        bob [gender="male"];
        eliza [gender="female"];
        fred [gender="male"];
        john [gender="male"];
        mary [gender="female"];
        steve [gender="male"];
        sue [gender="female"];
        mark [gender="male"];

        john -> mary;
        john -> bob;
        mary -> sue;
        sue -> bob;
        sue -> mary;
        fred -> bob;
        eliza -> steve;
}
$ gc talk.dot
       8       7 talks (talk.dot)
$
```

Count nodes with gvpr

```
$ gvpr 'N {clone($O, $)}' talk.dot # Clone each node to the output graph
digraph gvpr_result {
        bob     [gender=male];
        eliza   [gender=female];
        fred    [gender=male];
        john    [gender=male];
        mary    [gender=female];
        steve   [gender=male];
        sue     [gender=female];
        mark    [gender=male];
}
$
```

Node selection

```
$ gvpr 'N [gender == "female"] {clone($O, $)}' talk.dot # Select female nodes
digraph gvpr_result {
        eliza   [gender=female];
        mary    [gender=female];
        sue     [gender=female];
}
$
```

Edge selection

```
$ cat >male-talkers.gvpr <<\EOF
> BEGIN {
>         graph_t g = graph("new", "D")   // Initialize new directional graph
> }
>
> E [$.tail.gender == "male"] {
>         clone(g, $)                     // Clone edges with male talker
> }
>
> END {
```

```
>         write(g)                        // Write result
> }
> EOF
$ gvpr -f male-talkers.gvpr talk.dot
digraph new {
        fred     [gender=male];
        bob      [gender=male];
        fred -> bob;
        john     [gender=male];
        john -> bob;
        mary     [gender=female];
        john -> mary;
}
$
```

Node and edge properties

```
$ cat >color.gvpr <<\EOF
> N [gender == "female"] {color="pink"}   // Make female nodes pink
> N [gender == "male"] {color = "cyan"}  // Make male nodes cyan
> EOF
$ gvpr -c -f color.gvpr talk.dot |      # Copy graph applying changes
$ tee colored-talk.dot
digraph talks {
        bob      [color=cyan, gender=male];
        eliza    [color=pink, gender=female];
        steve    [color=cyan, gender=male];
        eliza -> steve;
        fred     [color=cyan, gender=male];
        fred -> bob;
        john     [color=cyan, gender=male];
        john -> bob;
        mary     [color=pink, gender=female];
        john -> mary;
        sue      [color=pink, gender=female];
        mary -> sue;
        sue -> bob;
        sue -> mary;
        mark     [color=cyan, gender=male];
}
$ dot -Nstyle=filled -Tpng -Gdpi=10 -Gsize=192,108\! -o talk.png colored-talk.dot
$
```

Counting

```
$ cat >count.gvpr <<\EOF
> E {
>         $.tail.talker = $.tail.talker + 1;      // Increment edge's talker role
>         $.head.listener = $.head.listener + 1   // Increment edge's listener role
> }
> EOF
$ gvpr -c -f count.gvpr talk.dot
digraph talks {
        bob      [gender=male, listener=3];
        eliza    [gender=female, talker=1];
        steve    [gender=male, listener=1];
        eliza -> steve;
        fred     [gender=male, talker=1];
        fred -> bob;
        john     [gender=male, talker=2];
        john -> bob;
        mary     [gender=female, listener=2, talker=1];
```

```
        john -> mary;
        sue      [gender=female, listener=1, talker=2];
        mary -> sue;
        sue -> bob;
        sue -> mary;
}
$
```

Obtain connected components

```
$ cat >john-secrets.gvpr <<\EOF
> BEG_G {
>         $tvtype = TV_fwd;               // Forward arc depth-first traversal
>         $tvroot = node($, "john");      // Start with John
> }
>
> N [$tvroot = NULL; 1]                   // Stop after this DFS; select node
>
> END_G {
>         induce($T);                     // Induce edges
>         write($T);                      // Write resulting graph
>         exit(0);                        // Do not process other graphs
> }
> EOF
$ gvpr -f john-secrets.gvpr talk.dot
digraph gvpr_result {
        bob      [gender=male];
        john     [gender=male];
        john -> bob;
        mary     [gender=female];
        john -> mary;
        sue      [gender=female];
        mary -> sue;
        sue -> bob;
        sue -> mary;
}
$
```

Topological sorting

```
$ cat clothing
shirt necktie
socks shoes
shirt jacket
necktie vest
underwear shirt
underwear trousers
trousers shoes
trousers belt
belt necktie
shirt vest
vest jacket
jacket coat
trousers coat
$ tsort clothing
socks
underwear
trousers
shirt
belt
shoes
necktie
```

```
vest
jacket
coat
$
```