```
def indexAdd(idx, a, b, c):
  adx = idx.get(a)
  if None == adx:
    adx = {}
    idx[a] = adx
  bdx = adx.get(b)
  if None == bdx:
    adx[b] = {c}
  else:
    bdx.add(c)
  return idx
```

```
class GraphIndexed:
  def __init__(self):
    self.eav = {}
    self.ave = {}
    self.vea = {}

  def add(self, e, a, v):
    self.eav = indexAdd(self.eav, e, a, v)
    self.ave = indexAdd(self.ave, a, v, e)
    self.vea = indexAdd(self.vea, v, e, a)
    return self

  def get(self, e, a, v):
    return lookups[simplify(e, a, v)](self, e, a, v)
```

```
def simplify(*args):
  return ‘’.join(
    map(lambda x: '0' if isinstance(x, Var) else ‘1',
    args))

lookups = {
  '000': lambda g, e, a, v: all(g.eav),
  '001': lambda g, e, a, v: double(g.vea, v),
  '010': lambda g, e, a, v: double(g.ave, a),
  '011': lambda g, e, a, v: single(g.ave, a, v),
  '100': lambda g, e, a, v: double(g.eav, e),
  '101': lambda g, e, a, v: single(g.vea, v, e),
  '110': lambda g, e, a, v: single(g.eav, e, a),
  '111': lambda g, e, a, v: exists(g.eav, e, a, v)}

def merge(a, b): return a + b
```