operators信息

- 一个`operators_config.yml`配置文件，记录operator信息
- 读入`operators_config.yml`为字典`allOps`：
  - `allOps`： `operators`配置文件的原始信息，用户可以进行维护的配置文件
  - `opGraph`： 记录算符之间可以组合关系的`Graph`
  - `opVarsLib`： 算符可选参数的枚举库

示例

`operators_config.yml`

```yaml
op1: # opName
    info: [2172, 1177]
    input:
      var0: # first-class argument
          A: ['Open', 'Close', 'High', 'Low']
          B: ['Volum', 'oi']
      var1: # second-class argument
          allTradeDate: ['allTradeDate']
      var2: # third-class argument
          window: [5, 10, 15,20,25,30,40]
          method: ['std', 'mean', 'min', 'max','median', 'percent']
    output:
      var0: ['Aprime', 'Bprime']
      var1:
    # add more attributes for this op, such as dimension, symmetry of input vars..

op2: # opName
    info: [2172, 1177] # 此字段用于前后两个算符接口匹配判断

    # DETAILS
    input:
      var0: # first-class argument
          A: ['Open', 'Close', 'High', 'Low']
          B: ['Volum','oi','Open', 'Close', 'High', 'Low']
      var1: # second-class argument
          allTradeDate: ['allTradeDate', 'hourMinute']
      var2: # third-class argument
          window: [5, 10, 15,20,25,30,40]
          method: ['std', 'mean', 'min', 'max','median', 'percent']
```

`allOps`：

```
allOps = utils.read_yml('opConfig_test',prnt=True)
```

```
fillna:
  info: [1172, 1177]
  input:
    var0:
      A: [Open, Close, High, Low]
op1:
  info: [2172, 1177]
  input:
    var0:
      A: [Open, Close, High, Low]
      B: [Volum, oi]
    var1:
      allTradeDate: [allTradeDate]
    var2:
      method: [std, mean, min, max, median, percent]
      window: [5, 10, 15, 20, 25, 30, 40]
  output:
    var0: [Aprime, Bprime]
    var1: null
op2:
  info: [2172, 1177]
  input:
    var0:
      A: [Open, Close, High, Low]
      B: [Volum, oi, Open, Close, High, Low]
    var1:
      allTradeDate: [allTradeDate, hourMinute]
    var2:
      method: [std, mean, min, max, median, percent]
      window: [5, 10, 15, 20, 25, 30, 40]
op3:
  info: [2172, 1177]
  input:
    var0:
      A: [Open, Close, High, Low]
      B: [Volum, oi, Open, Close, High, Low]
    var1:
      allTradeDate: [allTradeDate, hourMinute]
    var2:
      method: [std, mean, min, max, median, percent]
      window: [5, 10, 15, 20, 25, 30, 40]
```

`opGraph`：存储算符之间的匹配信息

```
opGraph = utils.opLib_to_opGraph(allOps)
opGraph
```

```
{'fillna': ['op3', 'op2', 'fillna', 'op1'],
 'op1': ['op2', 'op1', 'fillna', 'op3'],
 'op2': ['op3', 'op2', 'fillna', 'op1'],
 'op3': ['op3', 'fillna', 'op1', 'op2']}
```

生成图的方式通过函数`some_match_rule_for_opInfo(a, b)`控制, `a`, `b`为算符的`info`字段，为算符的缩略信息记录.

`opVarsLib`: 包含配置文件指定参数的枚举组合

```
opVarsLib = utils.opLib_to_opVarsLib(allOps)
opVarsLib['op1'][0:3]
```

```
[{'A': 'Open',
  'B': 'Volum',
  'allTradeDate': 'allTradeDate',
  'method': 'std',
  'window': 5},
 {'A': 'Open',
  'B': 'Volum',
  'allTradeDate': 'allTradeDate',
  'method': 'std',
  'window': 10},
 {'A': 'Open',
  'B': 'Volum',
  'allTradeDate': 'allTradeDate',
  'method': 'std',
  'window': 15}]
```

创建一棵`operator`树

```
root = AnyNode(name='fillna', A = 'Close')
print(RenderTree(root))
```

```
AnyNode(A='Close', name='fillna')
```

```
tree = utils.recursion_to_build_tree(deepcopy(root), 2, allOps, opGraph, opVarsLib)
print(RenderTree(tree))
```

```
AnyNode(A='Close', name='fillna')
└── AnyNode(A='Close', B='Close', allTradeDate='hourMinute', method='mean', name='op3',
window=30)
    ├── AnyNode(A='High', B='Volum', allTradeDate='hourMinute', method='mean',
name='op2', window=40)
    └── AnyNode(A='Open', B='Close', allTradeDate='allTradeDate', method='std',
name='op2', window=25)
```

```
formula = utils.tree_to_formula(tree)
formula
```

```
"fillna(op3(op2(High,Volum,allTradeDate='hourMinute', method='mean',
window=40),op2(Open,Close,allTradeDate='allTradeDate', method='std',
window=25),allTradeDate='hourMinute', method='mean', window=30))"
```

More complex tree

```
tree = utils.recursion_to_build_tree(deepcopy(root), 6, allOps, opGraph, opVarsLib)
print(RenderTree(tree))
```

```
AnyNode(A='Close', name='fillna')
└── AnyNode(A='Close', B='Volum', allTradeDate='hourMinute', method='max', name='op2',
window=25)
    ├── AnyNode(A='Open', B='Volum', allTradeDate='hourMinute', method='min',
name='op2', window=15)
    │   ├── AnyNode(A='High', name='fillna')
    │   └── AnyNode(A='Close', B='Close', allTradeDate='hourMinute', method='mean',
name='op3', window=20)
    └── AnyNode(A='Close', B='High', allTradeDate='hourMinute', method='max',
name='op2', window=25)
        ├── AnyNode(A='High', name='fillna')
        └── AnyNode(A='Open', B='Volum', allTradeDate='allTradeDate', method='max',
name='op2', window=5)
```

```
formula = utils.tree_to_formula(tree)
formula
```

```
"fillna(op2(op2(fillna(High),op3(Close,Close,allTradeDate='hourMinute', method='mean',
window=20),allTradeDate='hourMinute', method='min',
window=15),op2(fillna(High),op2(Open,Volum,allTradeDate='allTradeDate', method='max',
window=5),allTradeDate='hourMinute', method='max', window=25),allTradeDate='hourMinute',
method='max', window=25))"
```

生成多个表达式，按照指定复杂度

```
expr_list = utils.generate_N_expr_given_complexity(10,4,'opConfig_test')
```

```
Finish oplibrary!
--- Generate 10 formula !
```

```
list(expr_list)
```

```
["fillna(op2(fillna(High),op1(Close,Volum,allTradeDate='allTradeDate', method='std',
window=25),allTradeDate='hourMinute', method='std', window=10))",
 "fillna(op2(op2(Close,oi,allTradeDate='hourMinute', method='std',
window=20),op1(Close,oi,allTradeDate='allTradeDate', method='min',
window=5),allTradeDate='hourMinute', method='std', window=15))",
 "fillna(op3(fillna(Open),op1(Close,oi,allTradeDate='allTradeDate', method='median',
window=30),allTradeDate='allTradeDate', method='std', window=30))",
 "fillna(fillna(op1(op1(Low,oi,allTradeDate='allTradeDate', method='std',
window=40),op1(High,Volum,allTradeDate='allTradeDate', method='mean',
window=5),allTradeDate='allTradeDate', method='mean', window=15)))",
 "fillna(op1(op1(High,Volum,allTradeDate='allTradeDate', method='mean',
window=20),op3(Close,Volum,allTradeDate='allTradeDate', method='min',
window=30),allTradeDate='allTradeDate', method='mean', window=25))",
 "fillna(op3(op2(Open,High,allTradeDate='hourMinute', method='percent',
window=20),op1(High,Volum,allTradeDate='allTradeDate', method='percent',
window=20),allTradeDate='hourMinute', method='percent', window=20))",
 "fillna(fillna(op1(op2(Close,High,allTradeDate='hourMinute', method='std',
window=10),op1(Close,Volum,allTradeDate='allTradeDate', method='median',
window=40),allTradeDate='allTradeDate', method='std', window=40)))",
 "fillna(fillna(op1(fillna(High),op1(Close,Volum,allTradeDate='allTradeDate',
method='min', window=15),allTradeDate='allTradeDate', method='min', window=10)))",
 "fillna(op1(op3(Close,High,allTradeDate='allTradeDate', method='std',
window=10),op2(Open,Close,allTradeDate='allTradeDate', method='min',
window=5),allTradeDate='allTradeDate', method='min', window=10))",
 "fillna(op1(op3(High,Open,allTradeDate='allTradeDate', method='min',
window=30),op3(Close,Low,allTradeDate='allTradeDate', method='percent',
window=15),allTradeDate='allTradeDate', method='max', window=25))"]
```