In [19]:
```
import numpy as np
import pandas as pd
from scipy import stats
# set up panel data
pdata = pd.read_csv("Bank-7-no-diff.csv", index_col = ["ID", "Year"],
                   parse_dates = True)
pdata
# alternatively
# data = pd.read_csv("http://web.pdx.edu/~crkl/ceR/data/airline.txt", sep='\s+',
# Set data as panel data
# pdata = data.set_index(['I', 'T'], inplace=True)
#dataset is as same as bank-5.csv
```

Out[19]:

		SL	roe	ta	cc	ffr	lf
ID	**Year**						
BOKF	**2004-01-01**	1	0.135	13.603520	6.624590	2.156129	-1.955696
	2005-01-01	1	0.135	13.641222	6.663562	4.157097	-0.731402
	2006-01-01	1	0.130	13.667888	6.696201	5.238065	-0.060310
	2007-01-01	1	0.118	13.694235	6.709663	4.244516	-0.836372
	2008-01-01	1	0.079	14.614250	6.777514	0.155161	-11.634505
...	**...**	...	...	...	...	...	...
COLB	**2017-01-01**	20	0.077	15.307629	7.380595	1.301613	-0.087971
	2018-01-01	20	0.087	15.222481	7.442430	2.274194	0.013209
	2019-01-01	20	0.091	15.230382	7.492348	1.550968	0.028774
	2020-01-01	20	0.068	15.805986	7.635558	0.090000	-7.560211
	2021-01-01	20	0.085	15.981578	7.707004	0.079677	8.358281

360 rows × 6 columns

In [20]:
```
df = pdata
#df
```

In [23]:
```
X_vars = ['ta', 'cc', 'ffr', 'lf']
y_var = ['roe']
```

In [24]:

```
## Partial Correlation

import statsmodels.api as sm

residuals = {}
for y_var in df.keys():
    X_vars = list(df.keys())
    X_vars.remove(y_var)
    X = df[X_vars]
    # Initial estimate should include constant
    #   This won't be the case we regress the errors
    X["Constant"] = 1
    # pass y_var as list for consistent structure
    y = df[[y_var]]
    model = sm.OLS(y, X)
    results = model.fit()
    residuals[y_var] = results.resid
residuals = pd.DataFrame(residuals)
residuals
```

Out[24]:

```
                          SL        roe         ta         cc        ffr         lf
  ID        Year
BOKF  2004-01-01   -9.530257   0.014071  -0.321084   0.141459  -1.692158   9.177151
      2005-01-01   -9.539970   0.015682   0.000946  -0.012155   0.216152   1.247630
      2006-01-01   -9.511602   0.010978   0.162251  -0.086839   1.207758  -3.116871
      2007-01-01   -9.438416  -0.001159  -0.005984  -0.002394   0.266981   0.334795
      2008-01-01   -9.642460   0.032678   0.094591  -0.092246  -0.117051   6.524040
 ...         ...         ...        ...        ...        ...        ...        ...
COLB  2017-01-01    9.626385  -0.030012   0.033898  -0.005572   0.243639   1.220043
      2018-01-01    9.608442  -0.027036   0.020789   0.022275   0.521665  -4.373958
      2019-01-01    9.620138  -0.028976  -0.174292   0.128238  -0.444770  -2.532843
      2020-01-01    9.492490  -0.007808  -0.080337   0.082031  -0.122463  -6.253653
      2021-01-01    9.693317  -0.041111  -0.015721   0.019439  -0.225219   7.696303

360 rows × 6 columns
```

In [29]:

```
##DAG

import pingouin
from pgmpy.estimators import PC
import matplotlib.pyplot as plt
from matplotlib.patches import ArrowStyle
from networkx.drawing.nx_agraph import graphviz_layout
import warnings
warnings.filterwarnings("ignore")
from matplotlib.backends.backend_pdf import PdfPages
import networkx as nx
```

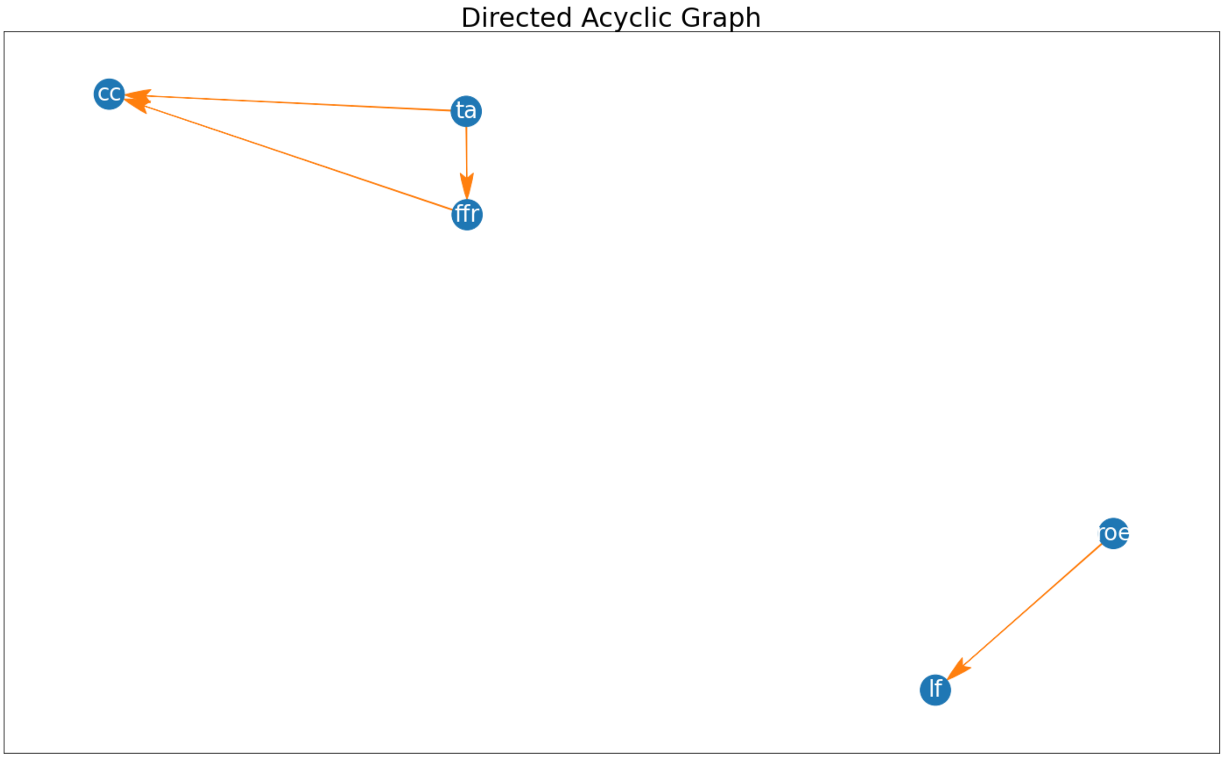
In [25]:

```
p_val = .01
from pgmpy.estimators import PC
c = PC(df)
max_cond_vars = len(df.keys())-2



model = c.estimate(return_type = "dag",variant= "parallel",#"orig", "stable"
                   significance_level = p_val,
                   max_cond_vars = max_cond_vars, ci_test = "pearsonr")
edges = model.edges()
```

```
Working for n conditional variables: 4:                                   4/4 [00:00<00:00,

100%                                                                      3.22it/s]
```

```
In [26]: from matplotlib.patches import ArrowStyle

         def graph_DAG(edges, df, title = ""):
             graph = nx.DiGraph()
             graph.add_edges_from(edges)
             color_map = ["C0" for g in graph]

             fig, ax = plt.subplots(figsize = (20,12))
             graph.nodes()
             plt.tight_layout()
             pos = nx.spring_layout(graph)#, k = 5/(len(sig_corr.keys())**.5))

             plt.title(title, fontsize = 30)
             nx.draw_networkx(graph, pos, node_color=color_map, node_size = 1200,
                              with_labels=True,  arrows=True,
                              font_color = "white",
                              font_size = 26, alpha = 1,
                              width = 1, edge_color = "C1",
                              arrowstyle=ArrowStyle("Fancy, head_length=3, head_width=1.5,

         graph_DAG(edges, df, title = "Directed Acyclic Graph")
         edges
```

```
Out[26]: OutEdgeView([('ta', 'cc'), ('ta', 'ffr'), ('roe', 'lf'), ('ffr', 'cc')])
```



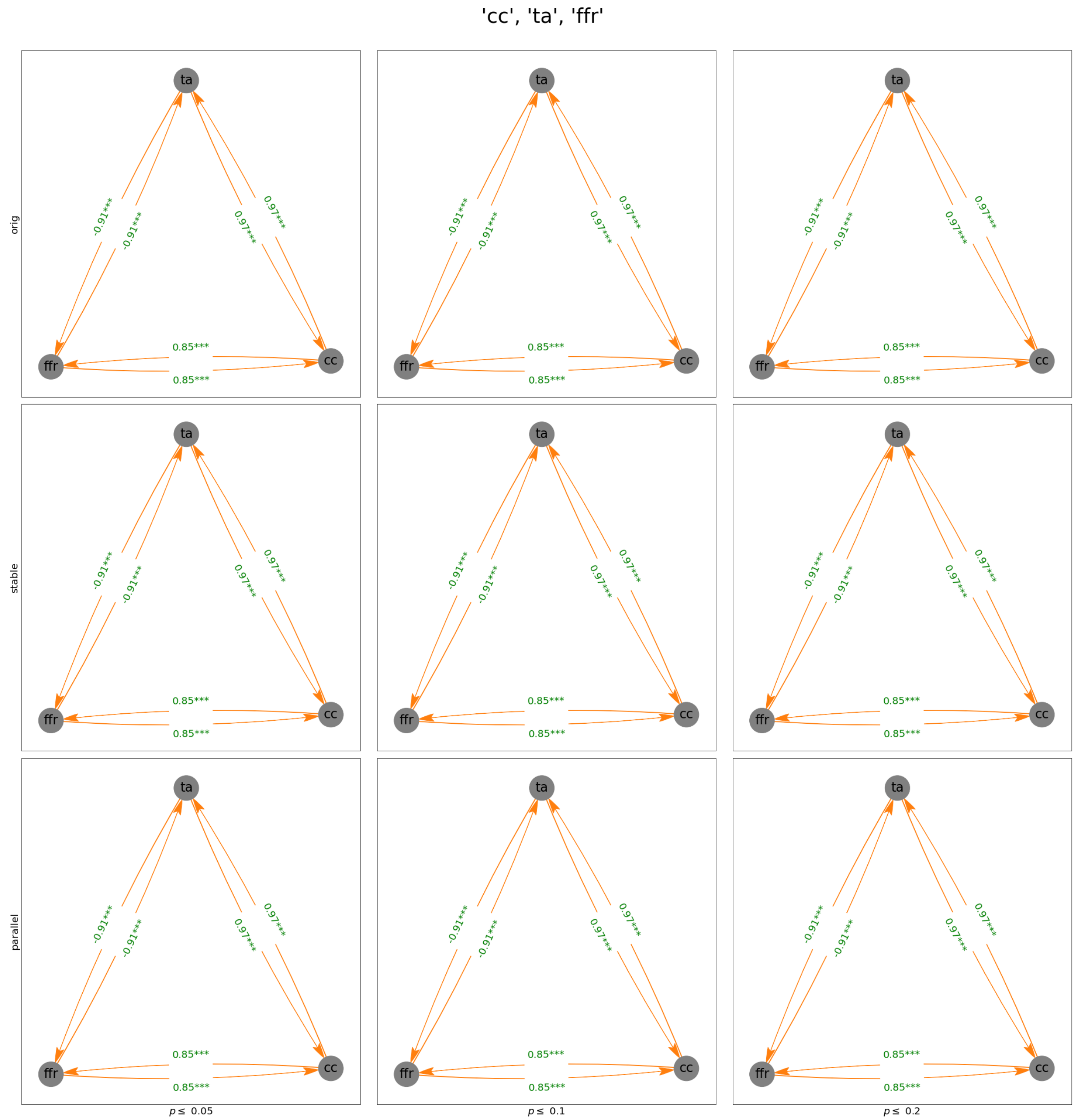
In [27]:

```
data = df
def firstLetterWord(str, num_chars = 3):

    result = ""

    # Traverse the string.
    v = True
    for i in range(len(str)):

        # If it is space, set v as true.
        if (str[i] == ' '):
            v = True

        # Else check if v is true or not.
        # If true, copy character in output
        # string and set v as false.
        elif (str[i] != ' ' and v == True):
            result += (str[i:i+num_chars])
            v = False

    return result
```

In [30]:
```
def graph_DAG(edges, data_reg, title = "",
              fig = False, ax = False,
              edge_labels = False,sig_vals = [0.05, 0.01, 0.001]):
    pcorr = data_reg.pcorr()
    graph = nx.DiGraph()
    def build_edge_labels(edges, df, sig_vals):
        edge_labels = {}
        for edge in edges:
            controls = [key for key in df.keys() if key not in edge]
            controls = list(set(controls))
            keep_controls = []
            for control in controls:
                control_edges = [ctrl_edge for ctrl_edge in edges if control == c
                if (control, edge[1]) in control_edges:
                    keep_controls.append(control)
#             print(edge, keep_controls)
            pcorr = df.partial_corr(x = edge[0], y = edge[1], covar=keep_controls
                                  method = "pearson")
            label = str(round(pcorr["r"][0],2))
            pvalue = pcorr["p-val"][0]
#             pcorr = df[[edge[0], edge[1]]+keep_controls].pcorr()
#             label = pcorr[edge[0]].loc[edge[1]]

            for sig_val in sig_vals:
                if pvalue < sig_val:
                    label = label + "*"

            edge_labels[edge] = label
        return edge_labels

    if edge_labels == False:
        edge_labels = build_edge_labels(edges,
                                        data_reg,
                                        sig_vals=sig_vals)
    graph.add_edges_from(edges)
    color_map = ["grey" for g in graph]

    if fig == False and ax == False: fig, ax = plt.subplots(figsize = (20,12))
    graph.nodes()
    plt.tight_layout()
    #pos = nx.spring_layout(graph)
    pos = graphviz_layout(graph)

    edge_labels2 = []
    for u, v, d in graph.edges(data=True):
        if pos[u][0] > pos[v][0]:
            if (v,u) in edge_labels.keys():
                edge_labels2.append(((u, v,), f'{edge_labels[u,v]}\n\n\n{edge_lab
        if (v,u) not in edge_labels.keys():
            edge_labels2.append(((u,v,), f'{edge_labels[(u,v)]}'))
    edge_labels = dict(edge_labels2)

    nx.draw_networkx(graph, pos, node_color=color_map, node_size = 2500,
                     with_labels=True,  arrows=True,
                     font_color = "black",
                     font_size = 26, alpha = 1,
                     width = 1, edge_color = "C1",
                     arrowstyle=ArrowStyle("Fancy, head_length=3, head_width=1.5,
                     connectionstyle='arc3, rad = 0.05',
                     ax = a)
    nx.draw_networkx_edge_labels(graph,pos,
                                 edge_labels=edge_labels,
                                 font_color='green',
                                 font_size=20,
                                ax = a)


DAG_models_vars = {0:["cc", "ta", "ffr"],
                  1:["roe", "cc", "ffr", ],
                  2:["roe", "ffr", "ta","lf"],
                  3:["ta", "roe", "ffr", "cc","lf"]}
# link_sigs = [0.05, 0.1, 0.2]
link_sigs = [0.05, .1, .2]
algorithms = ["orig", "stable", "parallel"]
for keys in DAG_models_vars.values():
    fig, ax = plt.subplots(len(algorithms), len(link_sigs), figsize = (30,30))
    max_cond_vars = len(keys) - 2
    data_reg = data[keys].dropna()
    data_reg.rename(columns = {col:firstLetterWord(col) for col in keys}, inplace
    keys = data_reg.keys()
    c = PC(data_reg[keys].dropna())
    max_cond_vars = len(keys) - 2
    i,j = 0,0
    for sig in link_sigs:
        for algorithm in algorithms:
            model = c.estimate(return_type = "pdag", variant = algorithm,
                               significance_level = sig,
                               max_cond_vars = max_cond_vars, ci_test = "pearsonr
            edges = model.edges()
            pcorr = data_reg.pcorr()
            weights = {}
            a = ax[i][j]
            graph_DAG(edges, data_reg, fig = fig, ax = a)

            if j == 0:
                a.set_ylabel(algorithm, fontsize = 20)
            if i == len(algorithms) - 1:
                a.set_xlabel("$p \leq$ "+ str(sig), fontsize = 20)
            i += 1
        j += 1
        i = 0
    plt.suptitle(str(list(keys)).replace("[","").replace("]",""), fontsize = 40,
    plt.show()
    plt.close()
edges
```

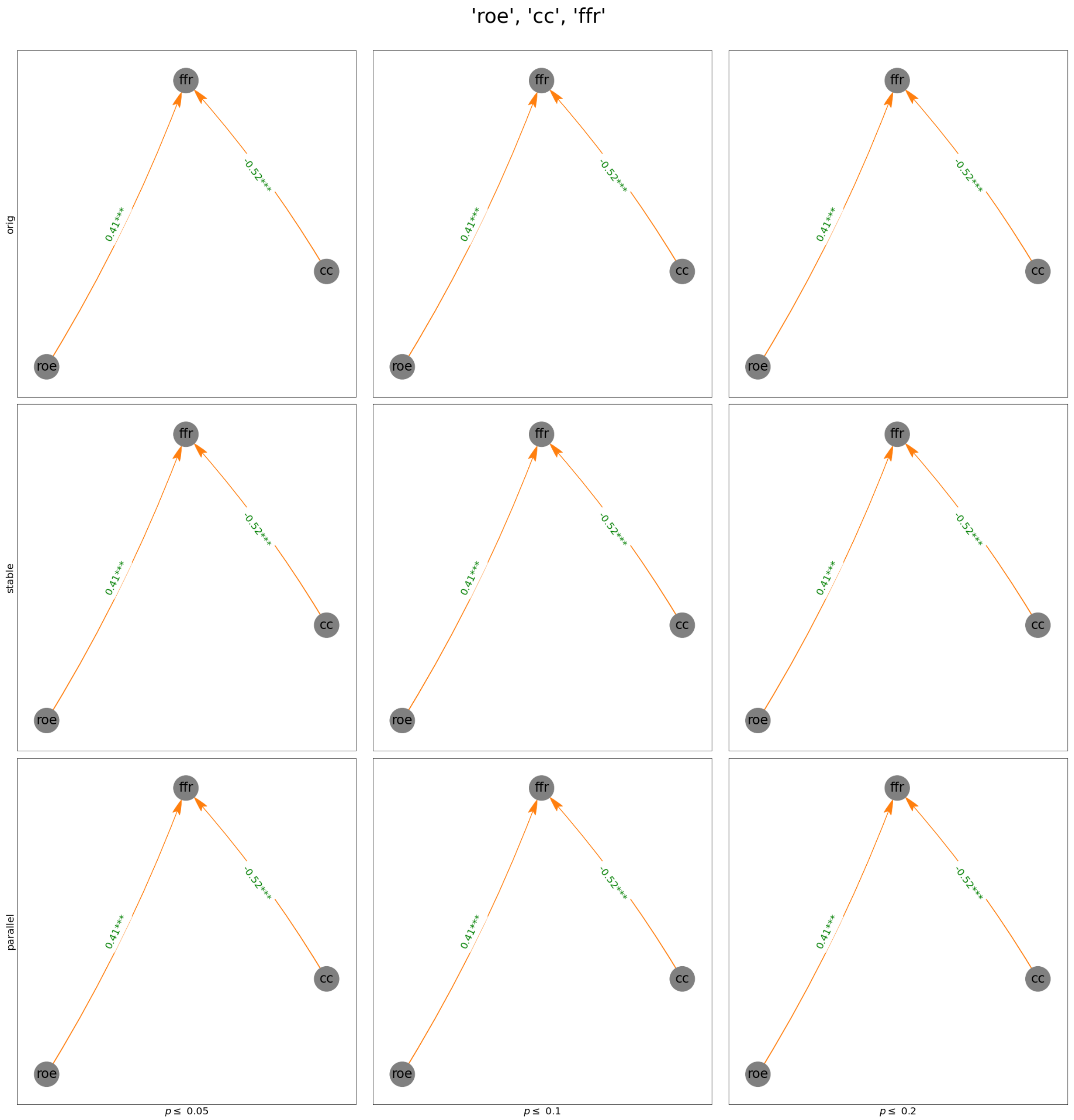
```
Working for n conditional variables:                          1/1 [00:00<00:00,
1: 100%                                                       11.89it/s]

Working for n conditional variables:                          1/1 [00:00<00:00,
1: 100%                                                       12.23it/s]
```

```
Working for n conditional variables: 1:                           1/1 [00:00<00:00,
100%                                                              7.63it/s]

Working for n conditional variables:                             1/1 [00:00<00:00,
1: 100%                                                          12.02it/s]

Working for n conditional variables:                             1/1 [00:00<00:00,
1: 100%                                                          14.04it/s]

Working for n conditional variables: 1:                           1/1 [00:00<00:00,
100%                                                              7.61it/s]

Working for n conditional variables:                             1/1 [00:00<00:00,
1: 100%                                                          10.21it/s]

Working for n conditional variables:                             1/1 [00:00<00:00,
1: 100%                                                          10.35it/s]

Working for n conditional variables:                             1/1 [00:00<00:00,
1: 100%                                                          10.85it/s]
```



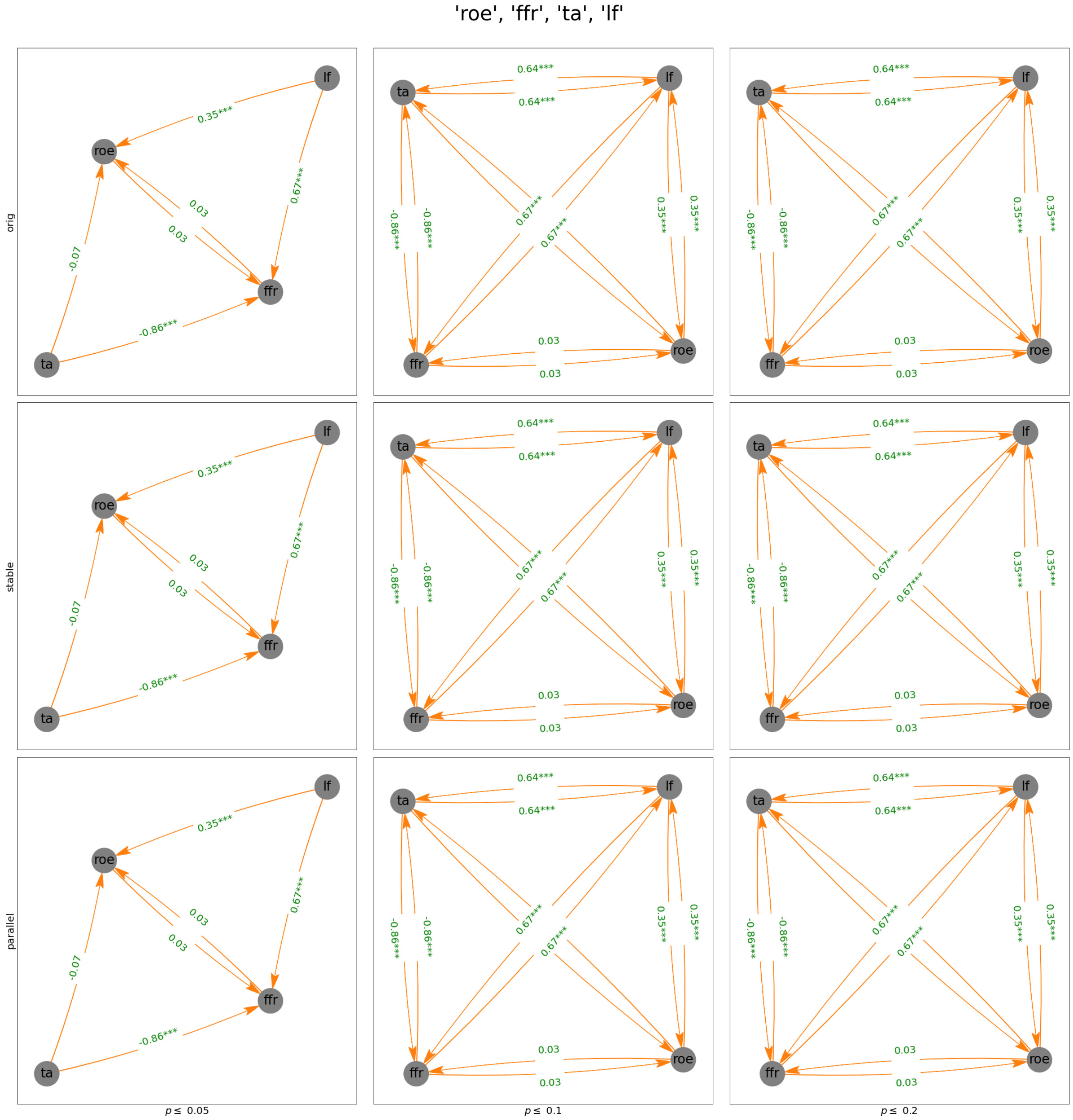
```
Working for n conditional variables:                    1/1 [00:00<00:00,
1: 100%                                                 16.45it/s]

Working for n conditional variables:                    1/1 [00:00<00:00,
1: 100%                                                 15.39it/s]

Working for n conditional variables: 1:                  1/1 [00:00<00:00,
100%                                                     8.51it/s]
```

```
Working for n conditional variables:                                  1/1 [00:00<00:00,
1: 100%                                                               16.41it/s]

Working for n conditional variables:                                  1/1 [00:00<00:00,
1: 100%                                                               18.09it/s]

Working for n conditional variables:                                  1/1 [00:00<00:00,
1: 100%                                                               14.46it/s]

Working for n conditional variables:                                  1/1 [00:00<00:00,
1: 100%                                                               22.89it/s]

Working for n conditional variables:                                  1/1 [00:00<00:00,
1: 100%                                                               20.10it/s]

Working for n conditional variables:                                  1/1 [00:00<00:00,
1: 100%                                                               14.32it/s]
```



```
Working for n conditional variables: 2:                    2/2 [00:00<00:00,
100%                                                       8.23it/s]

Working for n conditional variables:                      2/2 [00:00<00:00,
2: 100%                                                   14.21it/s]

Working for n conditional variables:                      2/2 [00:00<00:00,
2: 100%                                                   10.45it/s]

Working for n conditional variables: 2:                    2/2 [00:00<00:00,
100%                                                       8.36it/s]
```

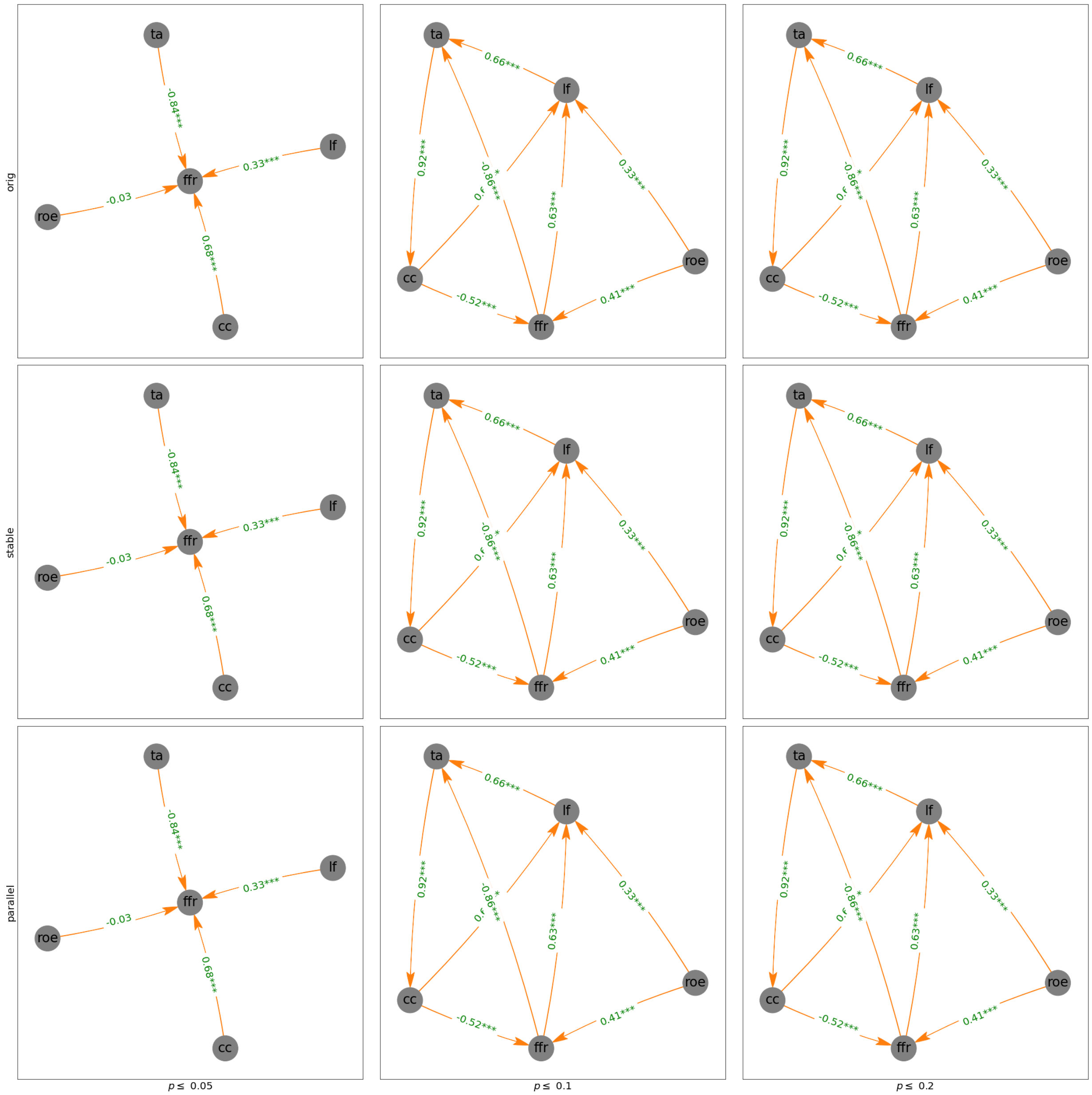
```
Working for n conditional variables: 2:                                2/2 [00:00<00:00,
100%                                                                   8.78it/s]

Working for n conditional variables: 2:                                2/2 [00:00<00:00,
100%                                                                   7.32it/s]

Working for n conditional variables:                                  2/2 [00:00<00:00,
2: 100%                                                               10.70it/s]

Working for n conditional variables:                                  2/2 [00:00<00:00,
2: 100%                                                               11.48it/s]

Working for n conditional variables: 2:                                2/2 [00:00<00:00,
100%                                                                   3.92it/s]
```



```
Working for n conditional variables: 3:                                 3/3 [00:00<00:00,

100%                                                                    4.76it/s]


Working for n conditional variables: 3:                                 3/3 [00:00<00:00,

100%                                                                    6.81it/s]


Working for n conditional variables: 3:                                 3/3 [00:00<00:00,

100%                                                                    5.38it/s]


Working for n conditional variables: 3:                                 3/3 [00:00<00:00,

100%                                                                    5.06it/s]


Working for n conditional variables: 3:                                 3/3 [00:00<00:00,

100%                                                                    4.68it/s]
```

```
Working for n conditional variables: 3:                         3/3 [00:00<00:00,
100%                                                            3.32it/s]

Working for n conditional variables: 3:                         3/3 [00:00<00:00,
100%                                                            5.13it/s]

Working for n conditional variables: 3:                         3/3 [00:00<00:00,
100%                                                            4.11it/s]

Working for n conditional variables: 3:                         3/3 [00:00<00:00,
100%                                                            3.62it/s]
```

'ta', 'roe', 'ffr', 'cc', 'lf'



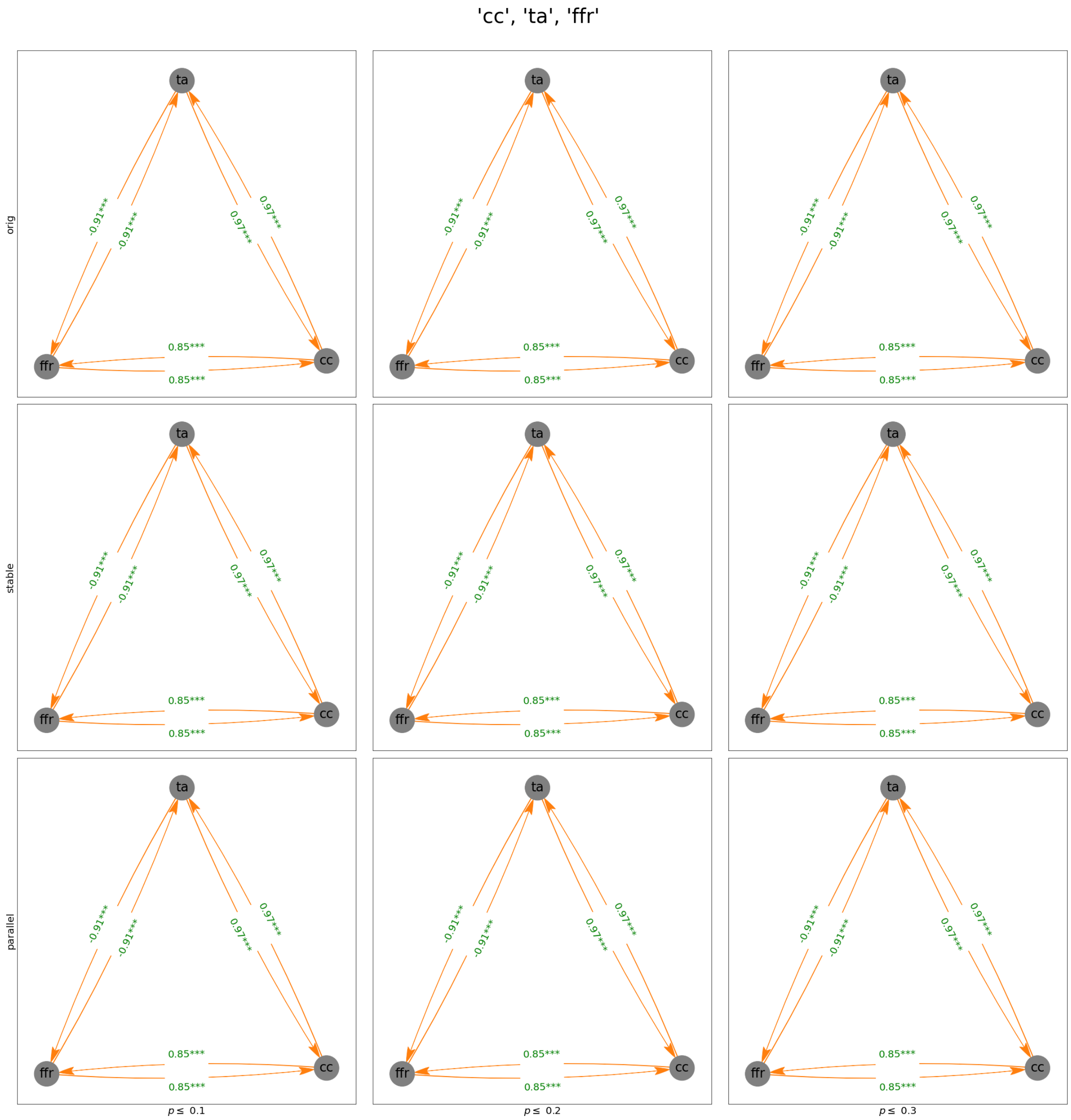
Out[30]:

```
OutEdgeView([('cc', 'lf'), ('cc', 'ffr'), ('lf', 'ta'), ('ffr', 'ta'), ('ffr',
'lf'), ('ta', 'cc'), ('roe', 'lf'), ('roe', 'ffr')])
```

In [31]:
```
def graph_DAG(edges, data_reg, title = "",
              fig = False, ax = False,
              edge_labels = False,sig_vals = [0.05, 0.01, 0.001]):
    pcorr = data_reg.pcorr()
    graph = nx.DiGraph()
    def build_edge_labels(edges, df, sig_vals):
        edge_labels = {}
        for edge in edges:
            controls = [key for key in df.keys() if key not in edge]
            controls = list(set(controls))
            keep_controls = []
            for control in controls:
                control_edges = [ctrl_edge for ctrl_edge in edges if control == c
                if (control, edge[1]) in control_edges:
                    keep_controls.append(control)
#             print(edge, keep_controls)
            pcorr = df.partial_corr(x = edge[0], y = edge[1], covar=keep_controls
                                  method = "pearson")
            label = str(round(pcorr["r"][0],2))
            pvalue = pcorr["p-val"][0]
#             pcorr = df[[edge[0], edge[1]]+keep_controls].pcorr()
#             label = pcorr[edge[0]].loc[edge[1]]

            for sig_val in sig_vals:
                if pvalue < sig_val:
                    label = label + "*"

            edge_labels[edge] = label
        return edge_labels

    if edge_labels == False:
        edge_labels = build_edge_labels(edges,
                                        data_reg,
                                        sig_vals=sig_vals)
    graph.add_edges_from(edges)
    color_map = ["grey" for g in graph]

    if fig == False and ax == False: fig, ax = plt.subplots(figsize = (20,12))
    graph.nodes()
    plt.tight_layout()
    #pos = nx.spring_layout(graph)
    pos = graphviz_layout(graph)

    edge_labels2 = []
    for u, v, d in graph.edges(data=True):
        if pos[u][0] > pos[v][0]:
            if (v,u) in edge_labels.keys():
                edge_labels2.append(((u, v,), f'{edge_labels[u,v]}\n\n\n{edge_lab
        if (v,u) not in edge_labels.keys():
            edge_labels2.append(((u,v,), f'{edge_labels[(u,v)]}'))
    edge_labels = dict(edge_labels2)

    nx.draw_networkx(graph, pos, node_color=color_map, node_size = 2500,
                     with_labels=True,  arrows=True,
                     font_color = "black",
                     font_size = 26, alpha = 1,
                     width = 1, edge_color = "C1",
                     arrowstyle=ArrowStyle("Fancy, head_length=3, head_width=1.5,
                     connectionstyle='arc3, rad = 0.05',
                     ax = a)
    nx.draw_networkx_edge_labels(graph,pos,
                                 edge_labels=edge_labels,
                                 font_color='green',
                                 font_size=20,
                                ax = a)


DAG_models_vars = {0:["cc", "ta", "ffr"],
                  1:["roe", "cc", "ffr", ],
                  2:["roe", "ffr", "ta","lf"],
                  3:["ta", "roe", "ffr", "cc","lf"]}
# link_sigs = [0.05, 0.1, 0.2]
link_sigs = [.1, .2, .3]
algorithms = ["orig", "stable", "parallel"]
for keys in DAG_models_vars.values():
    fig, ax = plt.subplots(len(algorithms), len(link_sigs), figsize = (30,30))
    max_cond_vars = len(keys) - 2
    data_reg = data[keys].dropna()
    data_reg.rename(columns = {col:firstLetterWord(col) for col in keys}, inplace
    keys = data_reg.keys()
    c = PC(data_reg[keys].dropna())
    max_cond_vars = len(keys) - 2
    i,j = 0,0
    for sig in link_sigs:
        for algorithm in algorithms:
            model = c.estimate(return_type = "pdag", variant = algorithm,
                               significance_level = sig,
                               max_cond_vars = max_cond_vars, ci_test = "pearsonr
            edges = model.edges()
            pcorr = data_reg.pcorr()
            weights = {}
            a = ax[i][j]
            graph_DAG(edges, data_reg, fig = fig, ax = a)

            if j == 0:
                a.set_ylabel(algorithm, fontsize = 20)
            if i == len(algorithms) - 1:
                a.set_xlabel("$p \leq$ "+ str(sig), fontsize = 20)
            i += 1
        j += 1
        i = 0
    plt.suptitle(str(list(keys)).replace("[","").replace("]",""), fontsize = 40,
    plt.show()
    plt.close()
edges
```

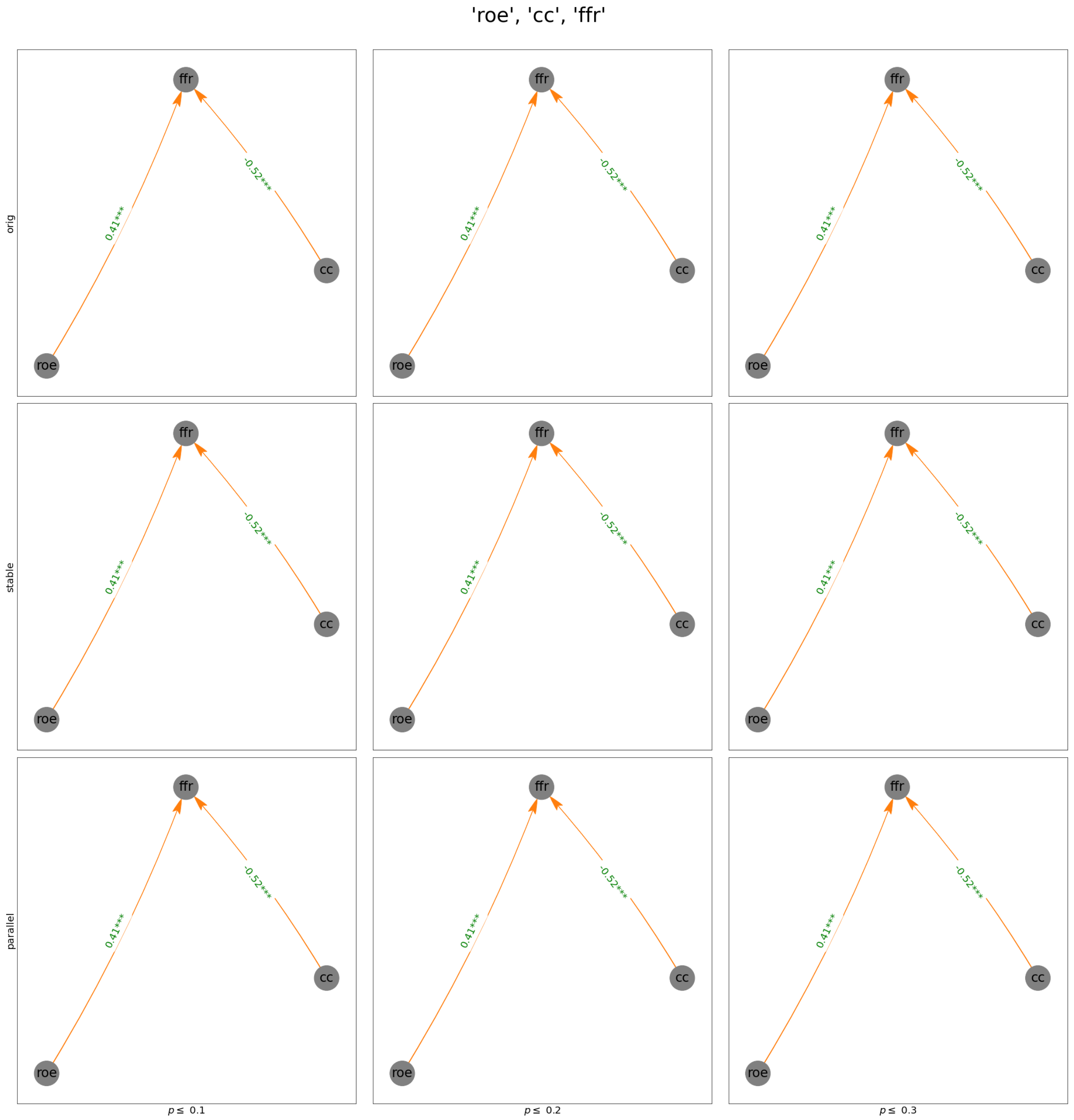
```
Working for n conditional variables: 1:                         1/1 [00:00<00:00,
100%                                                            8.70it/s]

Working for n conditional variables:                           1/1 [00:00<00:00,
1: 100%                                                        10.80it/s]
```

```
Working for n conditional variables: 1:                          1/1 [00:00<00:00,
100%                                                             8.02it/s]

Working for n conditional variables:                            1/1 [00:00<00:00,
1: 100%                                                         13.49it/s]

Working for n conditional variables:                            1/1 [00:00<00:00,
1: 100%                                                         11.33it/s]

Working for n conditional variables: 1:                          1/1 [00:00<00:00,
100%                                                             5.50it/s]

Working for n conditional variables:                            1/1 [00:00<00:00,
1: 100%                                                         12.23it/s]

Working for n conditional variables:                            1/1 [00:00<00:00,
1: 100%                                                         14.07it/s]

Working for n conditional variables: 1:                          1/1 [00:00<00:00,
100%                                                             9.69it/s]
```



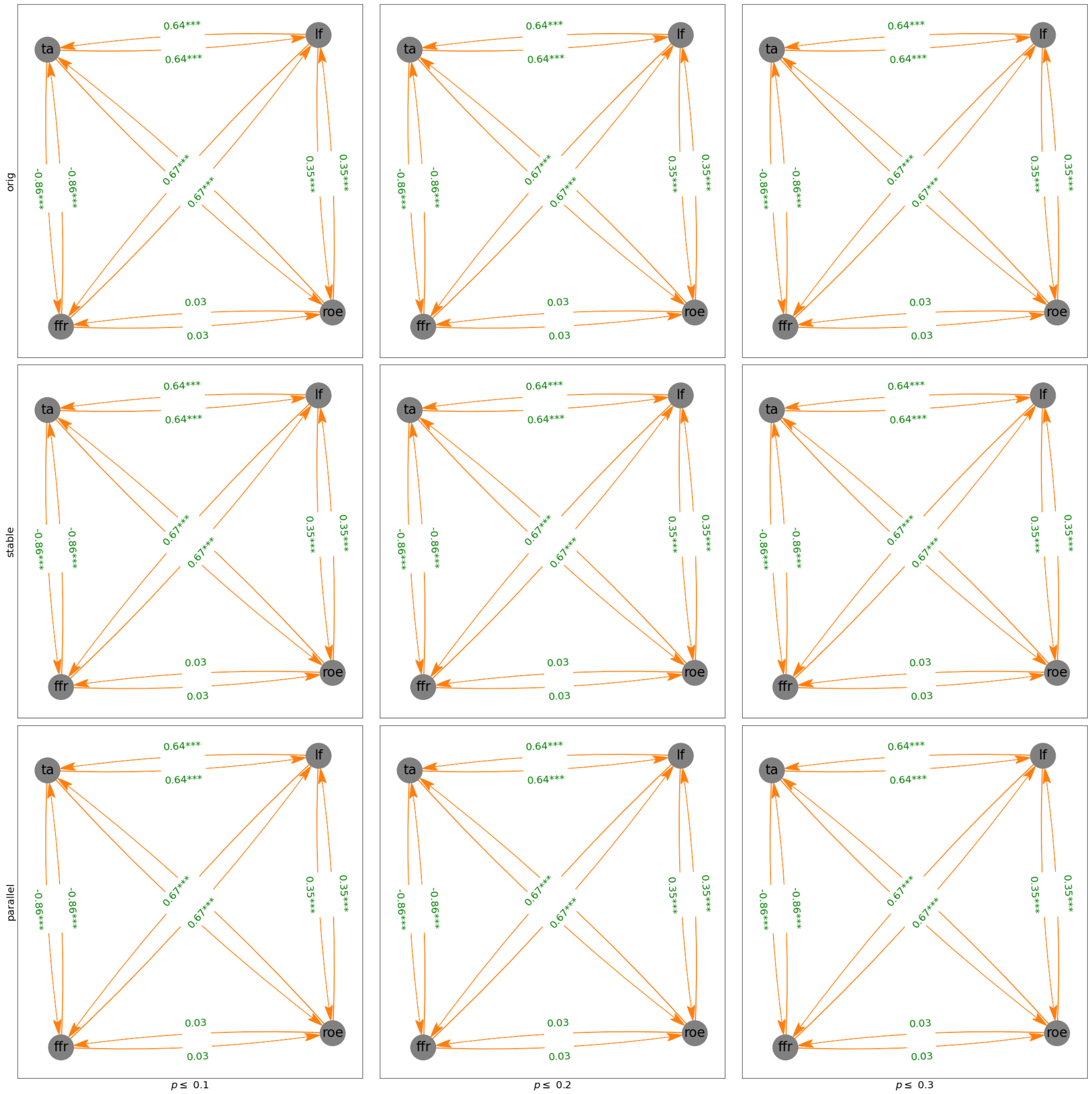
```
Working for n conditional variables:                         1/1 [00:00<00:00,
1: 100%                                                      23.55it/s]

Working for n conditional variables:                         1/1 [00:00<00:00,
1: 100%                                                      23.82it/s]

Working for n conditional variables:                         1/1 [00:00<00:00,
1: 100%                                                      13.96it/s]

Working for n conditional variables:                         1/1 [00:00<00:00,
1: 100%                                                      20.59it/s]

Working for n conditional variables:                         1/1 [00:00<00:00,
1: 100%                                                      20.35it/s]
```

```
Working for n conditional variables:                    1/1 [00:00<00:00,
1: 100%                                                 14.54it/s]

Working for n conditional variables:                  1/1 [00:00<00:00,
1: 100%                                               16.68it/s]

Working for n conditional variables:                    1/1 [00:00<00:00,
1: 100%                                                 15.63it/s]

Working for n conditional variables: 1:                  1/1 [00:00<00:00,
100%                                                     7.88it/s]
```



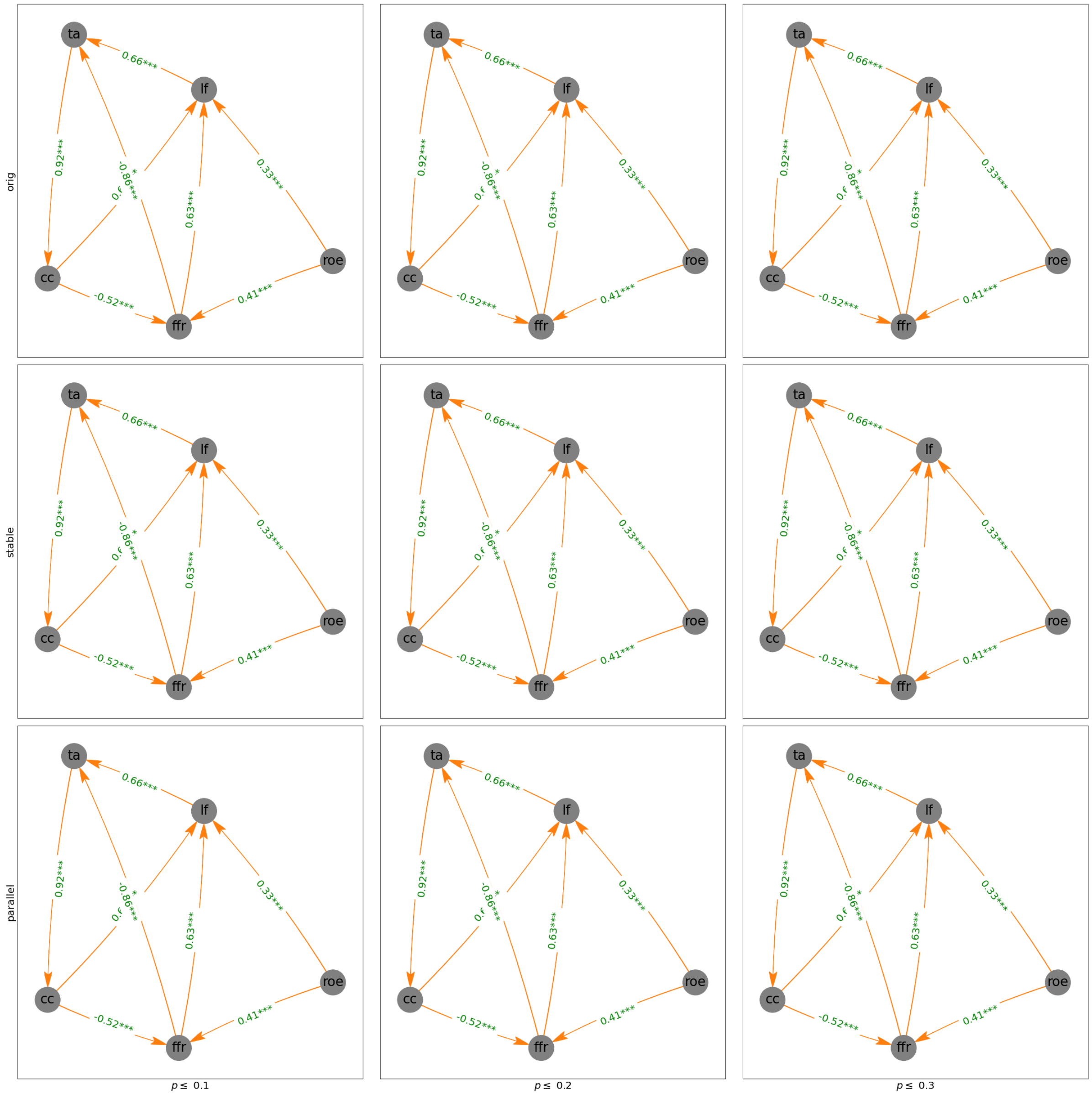
```
Working for n conditional variables: 2:              2/2 [00:00<00:00,
100%                                                 9.31it/s]

Working for n conditional variables: 2:              2/2 [00:00<00:00,
100%                                                 8.52it/s]

Working for n conditional variables: 2:              2/2 [00:00<00:00,
100%                                                 4.89it/s]

Working for n conditional variables:                2/2 [00:00<00:00,
2: 100%                                             10.19it/s]

Working for n conditional variables: 2:              2/2 [00:00<00:00,
100%                                                 8.60it/s]

Working for n conditional variables: 2:              2/2 [00:00<00:00,
100%                                                 4.36it/s]

Working for n conditional variables:                2/2 [00:00<00:00,
2: 100%                                             11.48it/s]

Working for n conditional variables:                2/2 [00:00<00:00,
2: 100%                                             12.82it/s]

Working for n conditional variables: 2:              2/2 [00:00<00:00,
100%                                                 8.84it/s]
```

'roe', 'ffr', 'ta', 'lf'


orig
stable
parallel
ta
lf
ffr
roe
0.64***
0.64***
-0.86***
-0.86***
0.67***
0.67***
0.35***
0.35***
0.03
0.03
$p \leq$ 0.1
$p \leq$ 0.2
$p \leq$ 0.3

```
Working for n conditional variables:                          3/3 [00:00<00:00,
3: 100%                                                       4.69it/s]

Working for n conditional variables: 3:                        3/3 [00:00<00:00,
100%                                                           5.05it/s]

Working for n conditional variables: 3:                        3/3 [00:00<00:00,
100%                                                           3.67it/s]

Working for n conditional variables: 3:                        3/3 [00:00<00:00,
100%                                                           5.10it/s]

Working for n conditional variables: 3:                        3/3 [00:00<00:00,
100%                                                           5.42it/s]

Working for n conditional variables: 3:                        3/3 [00:00<00:00,
100%                                                           3.33it/s]

Working for n conditional variables: 3:                        3/3 [00:00<00:00,
100%                                                           5.53it/s]

Working for n conditional variables: 3:                        3/3 [00:00<00:00,
100%                                                           5.53it/s]

Working for n conditional variables: 3:                        3/3 [00:00<00:00,
100%                                                           3.90it/s]
```

'ta', 'roe', 'ffr', 'cc', 'lf'



```
Out[31]: OutEdgeView([('cc', 'lf'), ('cc', 'ffr'), ('lf', 'ta'), ('ffr', 'ta'), ('ffr',
'lf'), ('ta', 'cc'), ('roe', 'lf'), ('roe', 'ffr')])
```