DrawPauliTransferEval

```
SetDirectory @ NotebookDirectory[];
Import["../Link/QuESTlink.m"];
```

Doc

```
? DrawPauliTransferEval
```

Symbol

DrawPauliTransferEval[pauliString, circuit] renders and returns a graph of the evaluation of 'circuit' when converted to a series of Pauli transfer maps, acting upon the given initial Pauli string.

DrawPauliTransferEval[data] renders the pre–computed evaluation graph 'data' as output by CalcPauliTransferEval[].

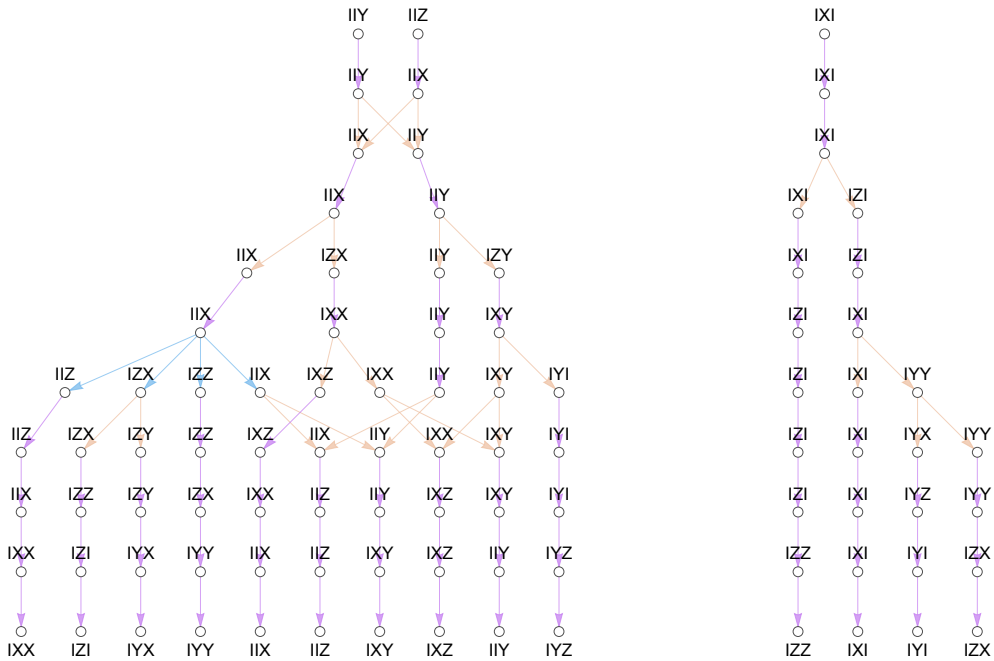
DrawPauliTransferEval accepts all options to Graph[], CalcPauliTransferEval[], DrawPauliTransferMap[], and some additional options, which we summarise below.

- "HighlightPathTo" –> pauliString (or a list of Pauli strings) highlights all edges ultimately contributing to the coefficient of the specified final pauliString(s). Symbolically weighted sums of Pauli strings are also accepted, in which case all edges to all non–orthogonal Pauli strings are highlighted.
- "CombineStrings" –> False disables combining incident Pauli strings so that the result is an (likely significantly larger) acyclic tree.
- "PauliStringForm" sets the vertex label format to one of "String", "Hidden" (these are the defaults depending on graph size), "Index", "Kronecker", or "Subscript". See ?GetPauliStringReformatted.
- "ShowCoefficients" –> True or False explicit shows or hides the PTMap coefficient associated with each edge. The default is Automatic which auto–hides edge labels if there are too many.
- "EdgeDegreeStyles" specifies the style of edges from nodes of increasing outdegree. See ?DrawPauliTransferMap.
- "CacheMaps" controls the automatic caching of generated PTMaps. See ?ApplyPauliTransferMap.
- AssertValidChannels –> False disables the simplification of symbolic Pauli string coefficients, only noticeable when "ShowCoefficients"–>True. See ?AssertValidChannels.
- Graph[] options override these settings. For example, specifying EdgeStyle –> Black will set all edges to Black regardless of their node's outdegree.

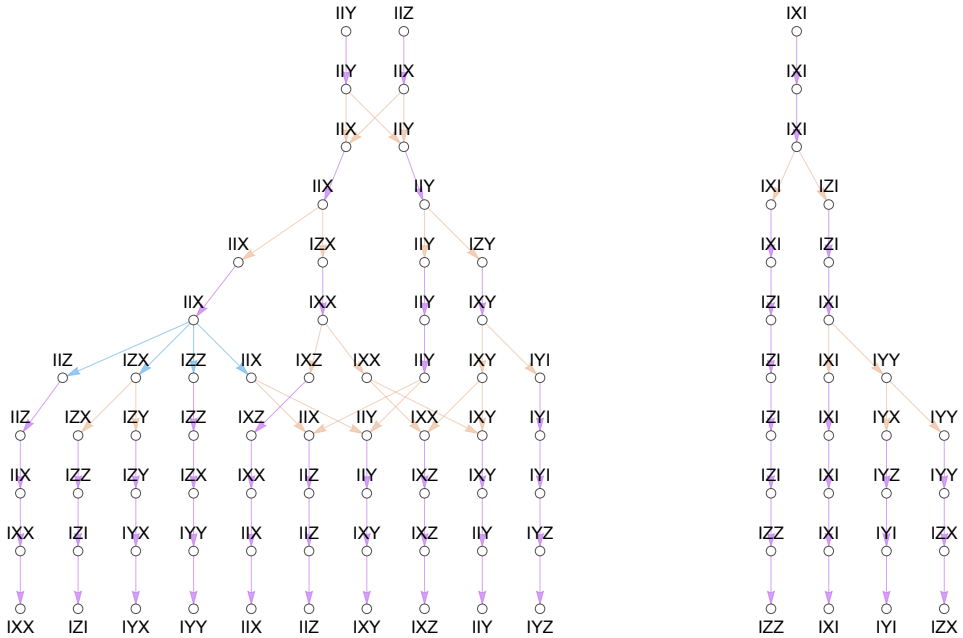
Correctness

Given evaluation data

```
u = Circuit[H0 Rz0[a] Ry1[b] Damp1[c] H1 C1[Ry0[a]] Ph0[x] H0 C0[X1] Z2];
in = Z0 + Y0 + X1;
data = CalcPauliTransferEval[in, u];
DrawPauliTransferEval[data]
```

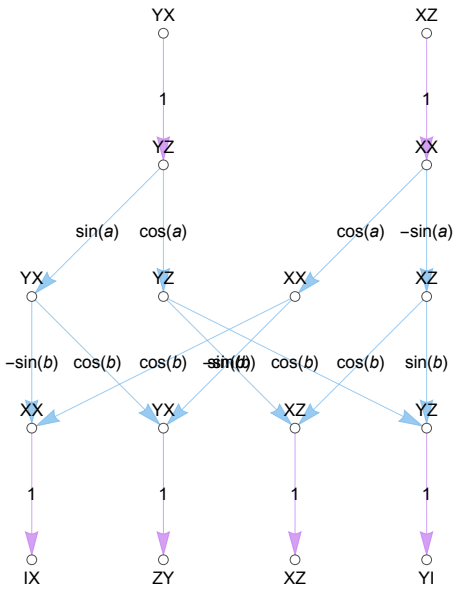


```
(* detailed data should yield an identical graph *)
data = CalcPauliTransferEval[in, u, "OutputForm" → "Detailed"];
DrawPauliTransferEval[data]
```

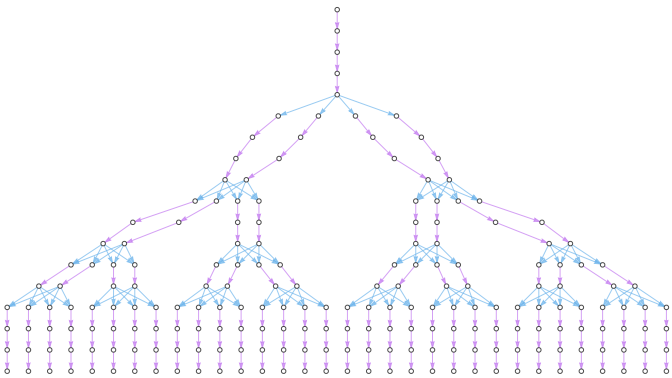


Given circuit

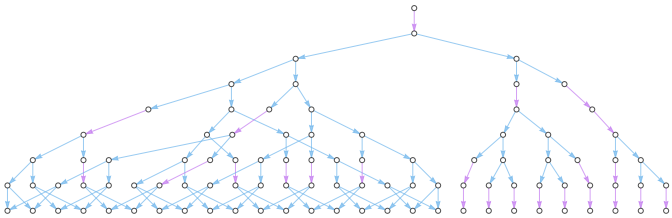
```
u = Circuit[H0 Ry0[a] Rz1[b] C0[X1]];
DrawPauliTransferEval[X1 Z0 + Y1 X0, u]
```



```
u = GetKnownCircuit["QFT", 5];
DrawPauliTransferEval[X0, u]
```

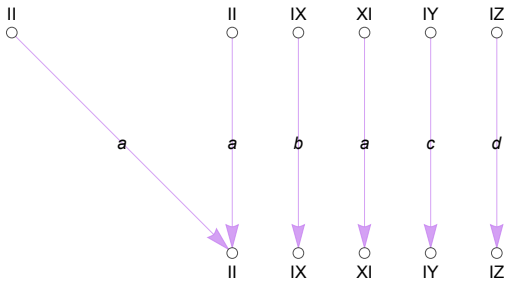


```
h = GetRandomPauliString[4, 8];
u = GetKnownCircuit["Trotter", h, 1, 1, π];
DrawPauliTransferEval[X0 Y1 Z2, u]
```

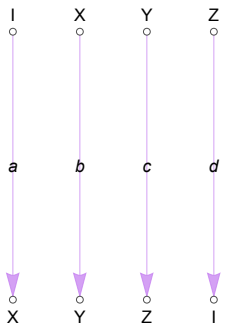


Given PTMs and PTMaps

```
u = { PTM0@DiagonalMatrix@{a, b, c, d}};
DrawPauliTransferEval[Id0 + X0 + Y0 + Z0 + Id1 + X1, u]
```



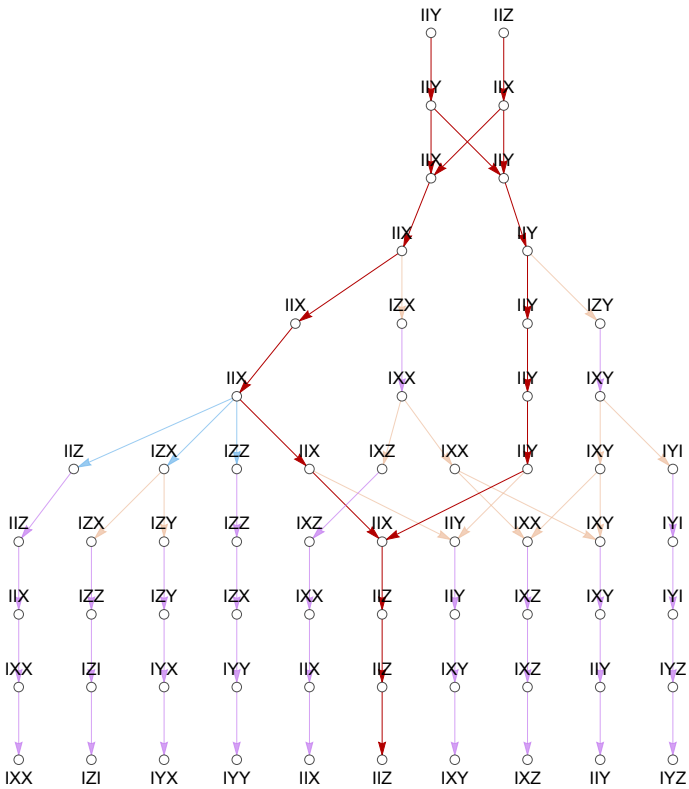
```
u = PTMap0[0 → {{1, a}}, 1 → {{2, b}}, 2 → {{3, c}}, 3 → {{0, d}}];
DrawPauliTransferEval[Id0 + X0 + Y0 + Z0, {u}]
```



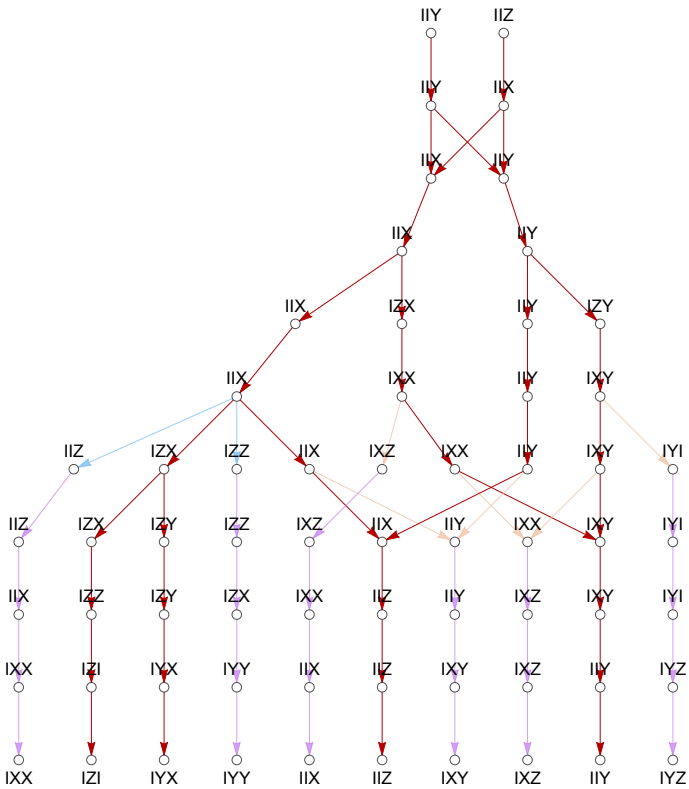
Options

“HighlightPathTo”

```
u = Circuit[H0 Rz0[a] Ry1[b] Damp1[c] H1 C1[Ry0[a]] Ph0[x] H0 C0[X1] Z2];
DrawPauliTransferEval[Z0 + Y0, u, "HighlightPathTo" → Z0]
```



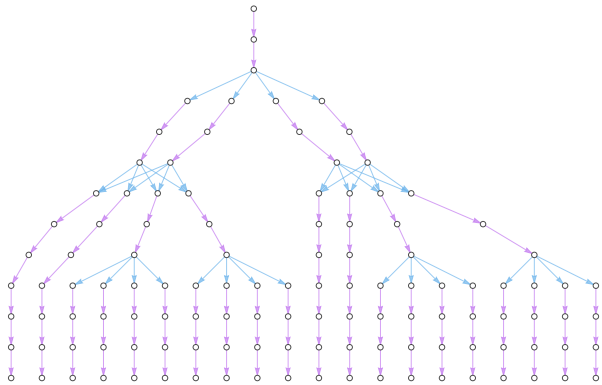
```
DrawPauliTransferEval[Z0 + Y0, u,
    "HighlightPathTo" → {Z0, Y0, a Z1 + b X0 Y1}]
```

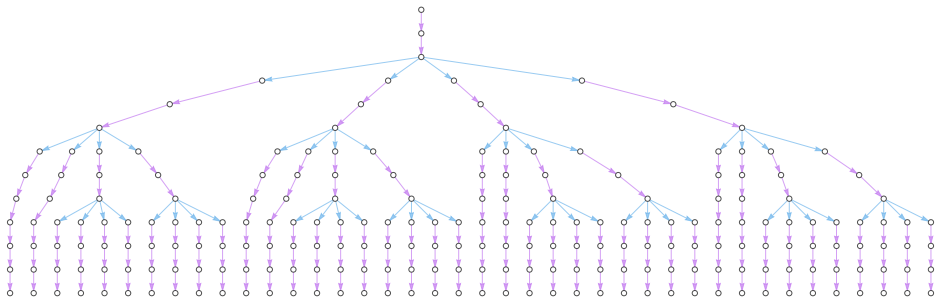


“CombineStrings”

```
u = GetKnownCircuit["QFT", 4];
```

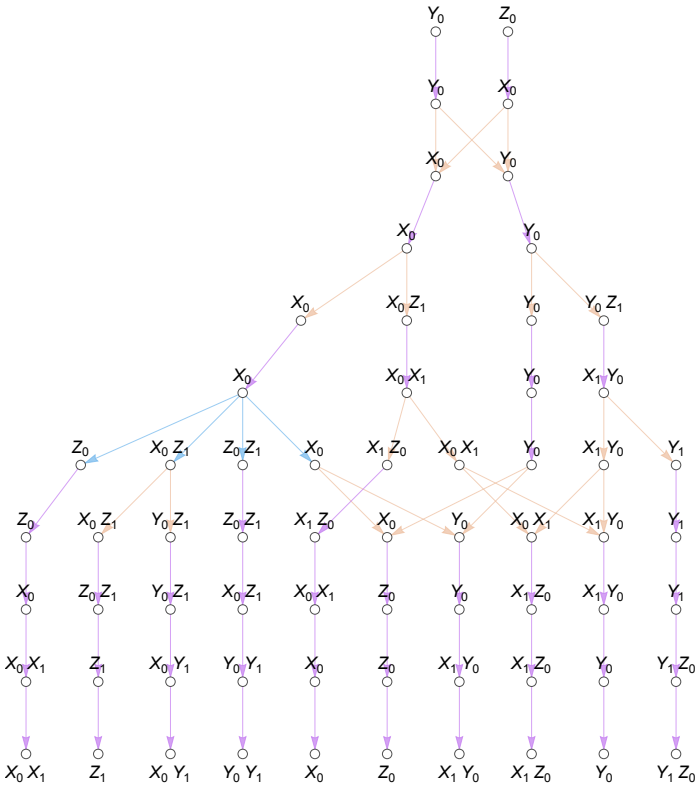
```
DrawPauliTransferEval[Z0 X1, u]
DrawPauliTransferEval[Z0 X1, u, "CombineStrings" → False]
```



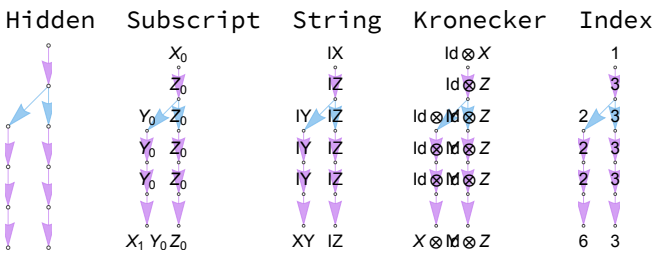


“PauliStringForm”

```
u = Circuit[H0 Rz0[a] Ry1[b] Damp1[c] H1 C1[Ry0[a]] Ph0[x] H0 C0[X1] Z2];

DrawPauliTransferEval[Z0 + Y0, u, "PauliStringForm" → "Subscript"]
```



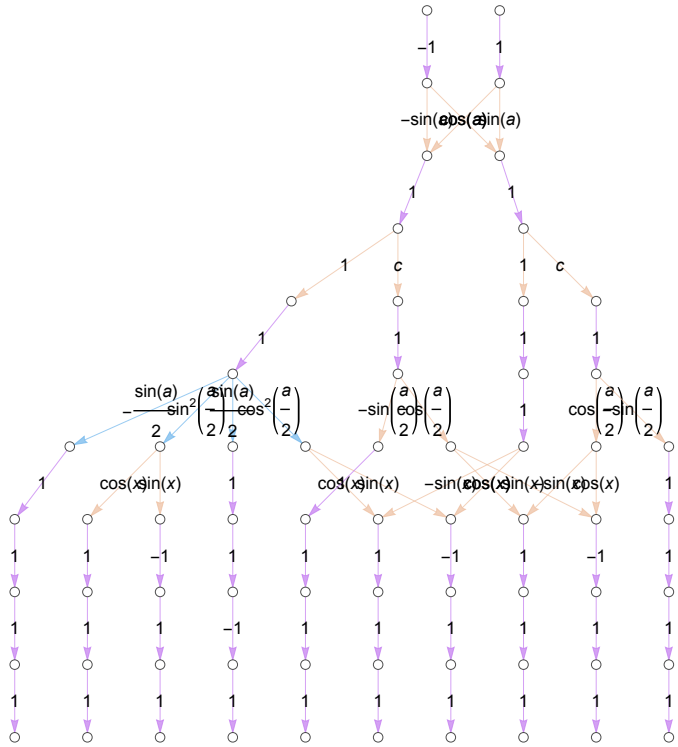
```
u = Circuit[H0 Rx0[a] Damp0[b] Depol0,1[c] C0[X1]];
options = {"Hidden", "Subscript", "String", "Kronecker", "Index"};

Row @ Riffle[Table[
      Column@{form,
            DrawPauliTransferEval[X0, u,
                  "PauliStringForm" → form,
                  "ShowCoefficients" → False]},
      {form, options}], "  "]
```



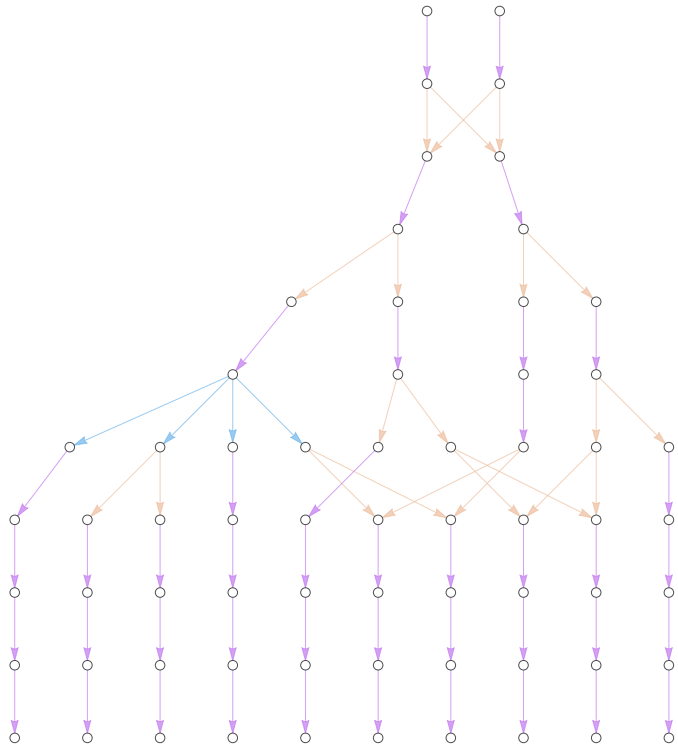
“ShowCoefficients”

```
u = Circuit[H0 Rz0[a] Ry1[b] Damp1[c] H1 C1[Ry0[a]] Ph0[x] H0 C0[X1] Z2];
```

```
DrawPauliTransferEval[Z0 + Y0, u,
    "ShowCoefficients" → True,
    "PauliStringForm" → "Hidden"]
```

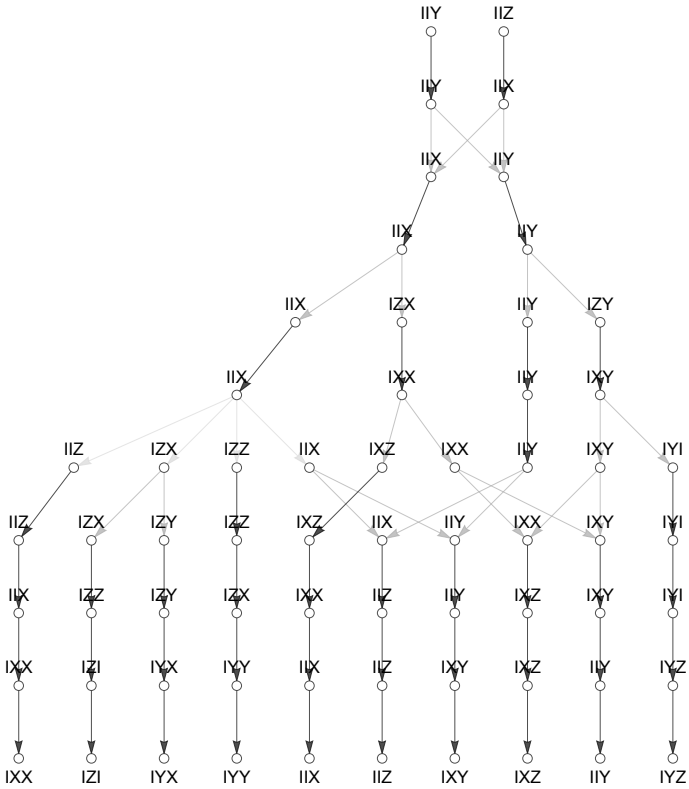


```
DrawPauliTransferEval[Z0 + Y0, u,
    "ShowCoefficients" → False,
    "PauliStringForm" → "Hidden"]
```

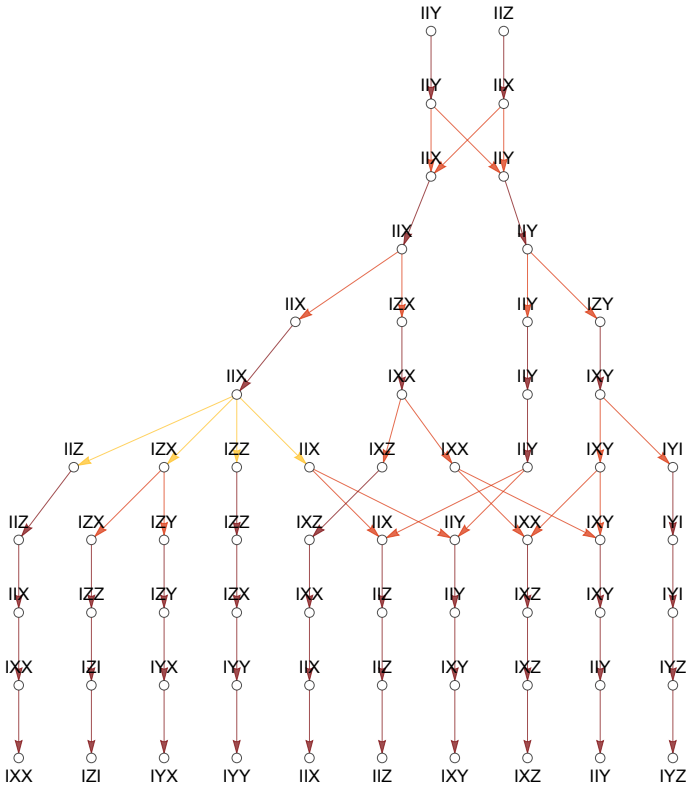


“EdgeDegreeStyles”

```
u = Circuit[H0 Rz0[a] Ry1[b] Damp1[c] H1 C1[Ry0[a]] Ph0[x] H0 C0[X1] Z2];
DrawPauliTransferEval[Z0 + Y0, u,
    "EdgeDegreeStyles" → {Black, Lighter@Gray, White, LightGray}]
```



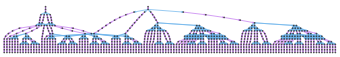
```
DrawPauliTransferEval[Z0 + Y0, u,
    "EdgeDegreeStyles" → ColorData["SolarColors"] /@ Range[0, 1, .3]]
```



“CacheMaps”

```
u = GetKnownCircuit["QFT", 5];
Timing @ DrawPauliTransferEval[X3 Z2 + X2, u]
Timing @ DrawPauliTransferEval[X3 Z2 + X2, u, "CacheMaps" → "Never"]
```

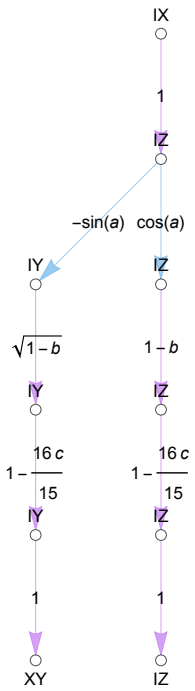
{1.29229,  }

{2.21091,  }

AssertValidChannels

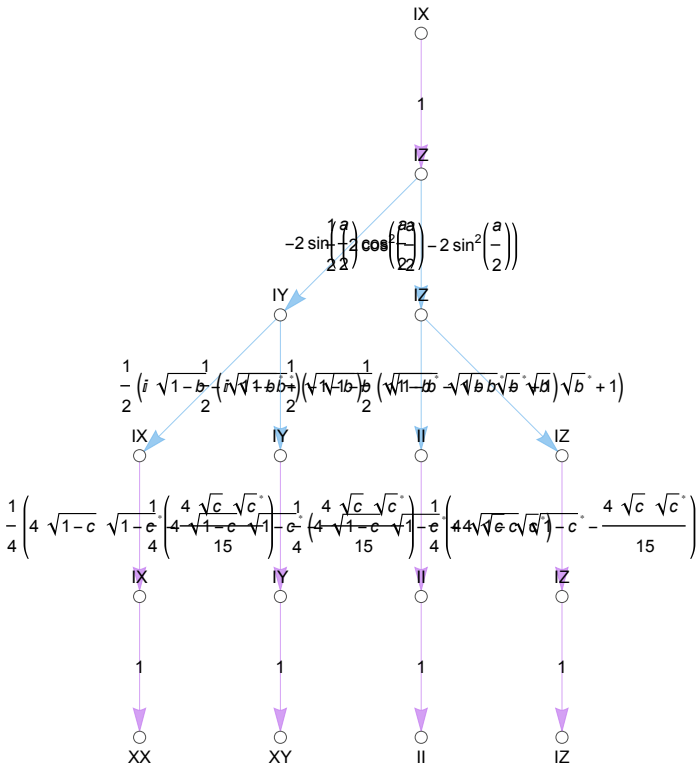
```
u = Circuit[H0 Rx0[a] Damp0[b] Depol0,1[c] C0[X1]];
```

```
DrawPauliTransferEval[X0, u]
```



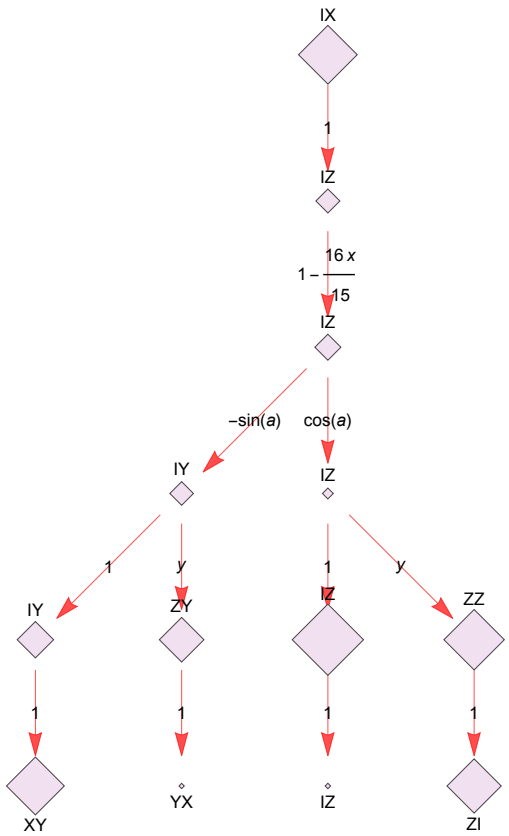

```
(* disabling simplification can cause zero weight branches to survive *)
```

```
DrawPauliTransferEval[X0, u, AssertValidChannels → False]
```

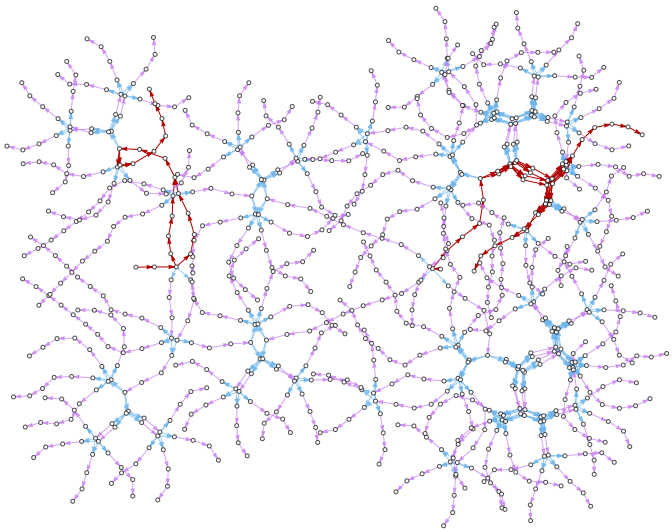


Graph[] options

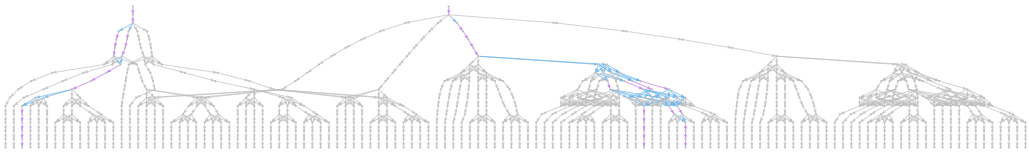
```
circ = Circuit[H0 Depol0,1[x] Rx0[a] Damp1[y] C0[X1]];
DrawPauliTransferEval[X0, circ,

    EdgeStyle → Red,
    VertexShapeFunction → "Diamond",
    VertexSize :→ .5 RandomReal[],
    VertexStyle → LightPurple
]
```



```
u = GetKnownCircuit["QFT", 5];

DrawPauliTransferEval[X3 Z2 + X2, u,
    "HighlightPathTo" → {X2, Y1 Y2, X4 X3 X2 Y1},
    GraphLayout → "SpringEmbedding"
]
```



```
DrawPauliTransferEval[X3 Z2 + X2, u,
    "HighlightPathTo" → {X2, Y1 Y2, X4 X3 X2 Y1},
    GraphHighlightStyle → "DehighlightGray"
]
```



Errors

```
DrawPauliTransferEval[X0, X0, "CombineStrings" → Eh]
```

DrawPauliTransferEval : Option "CombineStrings" must be True or False. See ?CalcPauliTransferEval.

```
$Failed
```

```
DrawPauliTransferEval[X0, X0, "CacheMaps" → Eh]
```

DrawPauliTransferEval : Option "CacheMaps" must be one of "Forever", "UntilCallEnd" or "Never". See ?ApplyPauliTransferMap.

```
$Failed
```

```
DrawPauliTransferEval[X0, X0, "ShowCoefficients" → Eh]
```

DrawPauliTransferEval : Option "ShowCoefficients" must be Automatic, True or False. See ?DrawPauliTransferEval .

```
$Failed
```

```
DrawPauliTransferEval[X0, X0, "PauliStringForm" → "Unknown"]
```

DrawPauliTransferEval : Invalid value for option "PauliStringForm". See ?DrawPauliTransferEval .

```
$Failed
```

```
DrawPauliTransferEval[X0, X0, "HighlightPathTo" → Eh]
DrawPauliTransferEval[X0, X0, "HighlightPathTo" → X-1]
```

DrawPauliTransferEval : Invalid value for option "HighlightPathTo ". See ?DrawPauliTransferEval .

```
$Failed
```

DrawPauliTransferEval : Invalid value for option "HighlightPathTo ". See ?DrawPauliTransferEval .

```
$Failed
```

```
DrawPauliTransferEval[X0, X0, "UnrecognisedOption" → Eh]
```

OptionValue : Unknown option UnrecognisedOption for {DrawPauliTransferEval , CalcPauliTransferEval , ApplyPauliTransferMap , CalcPauliTransferMap , Graph}.

```
$Failed
```

```
DrawPauliTransferEval[X-1, {X0}]
```

DrawPauliTransferEval : Invalid arguments. See ?DrawPauliTransferEval

```
$Failed
```

```
DrawPauliTransferEval[X0 Y0, {X0}]
```

DrawPauliTransferEval : Invalid arguments. See ?DrawPauliTransferEval

```
$Failed
```

```
DrawPauliTransferEval[{X0}, {}]
DrawPauliTransferEval[{}, {X0}]
DrawPauliTransferEval[]
```

DrawPauliTransferEval : Invalid arguments. See ?DrawPauliTransferEval

```
$Failed
```

DrawPauliTransferEval : Invalid arguments. See ?DrawPauliTransferEval

```
$Failed
```

DrawPauliTransferEval : Invalid arguments. See ?DrawPauliTransferEval

```
$Failed
```