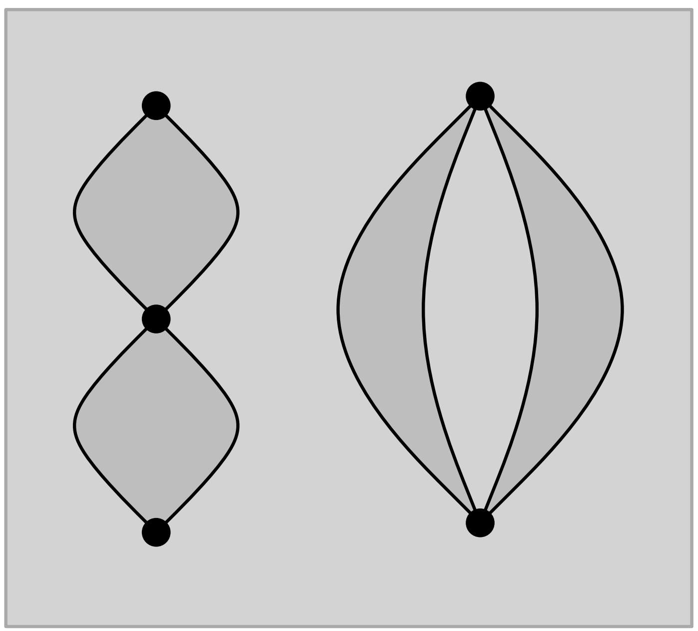
Karlsruhe Institute of Technology

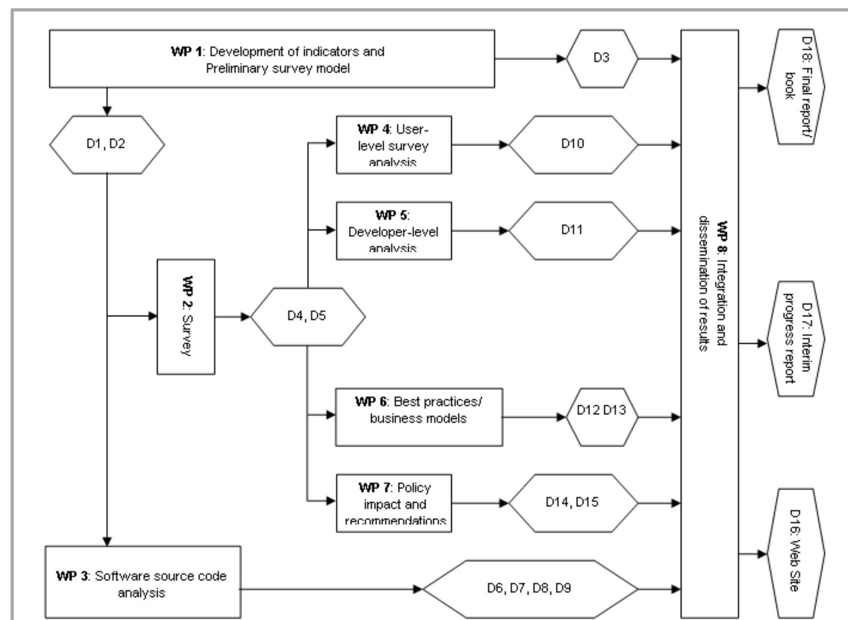
Algorithms for graph visualization

Divide and Conquer - Series-Parallel Graphs

WINTER SEMESTER 2013/2014

Tamara Mchedlidze – MARTIN NÖLLENBURG

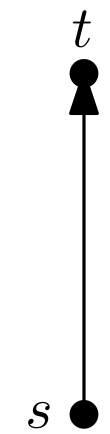




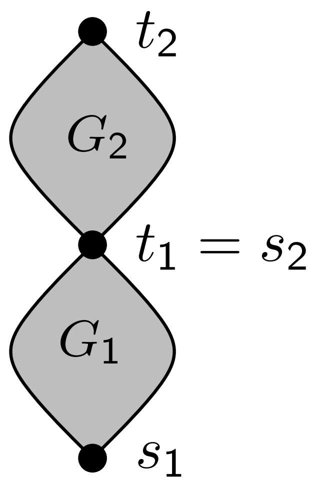
Series-parallel Graphs

Graph G is **series-parallel**, if

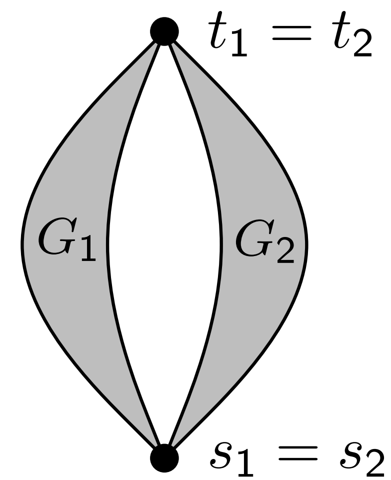
- It contains a single edge (s, t) (s-source, t-sink)
- It consists of two series-parallel graphs G_1, G_2 with sources s_1, s_2 and sinks t_1, t_2 which are combined using one of the following rules:



Series composition:
Identify t_1 and s_2,
s_1 is the source of G, t_2 is the sink of G



Parallel composition:
Identify s_1, s_2 and set it to be source of G
Identify t_1, t_2 and set it to be sink of G



Series-parallel Graphs. Decomposition Tree.

Karlsruhe Institute of Technology

Lemma

Series-parallel graphs are acyclic and planar.

In order to proof this statement we can use a **decomposition tree** of G, which is a binary tree T with nodes of three types: S,P and Q-type.

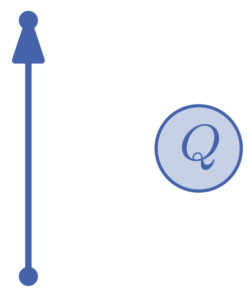
Series-parallel Graphs. Decomposition Tree.

Lemma

Series-parallel graphs are acyclic and planar.

In order to proof this statement we can use a **decomposition tree** of G, which is a binary tree T with nodes of three types: S,P and Q-type.

- If G is a single edge, then the corresponding node is Q-node



Series-parallel Graphs. Decomposition Tree.

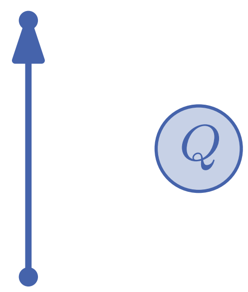
Karlsruhe Institute of Technology

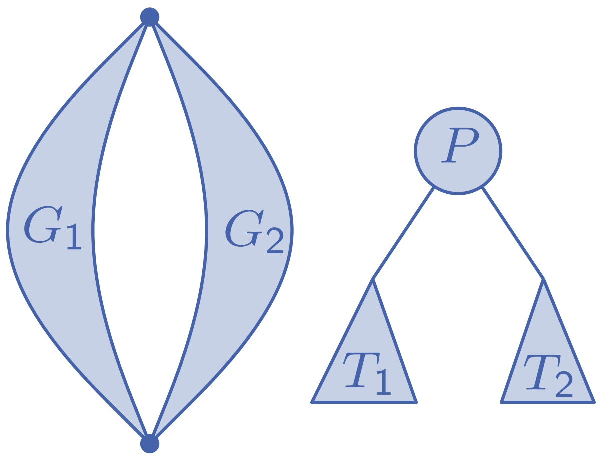
Lemma

Series-parallel graphs are acyclic and planar.

In order to proof this statement we can use a **decomposition tree** of G, which is a binary tree T with nodes of three types: S,P and Q-type.

- If G is a single edge, then the corresponding node is Q-node
- If G is a parallel composition of G_1 (with tree T_1) and G_2 (with tree T_2), then the root of T is P-node and T_1 is its left subtree, T_2 is its right subtree





Series-parallel Graphs. Decomposition Tree.

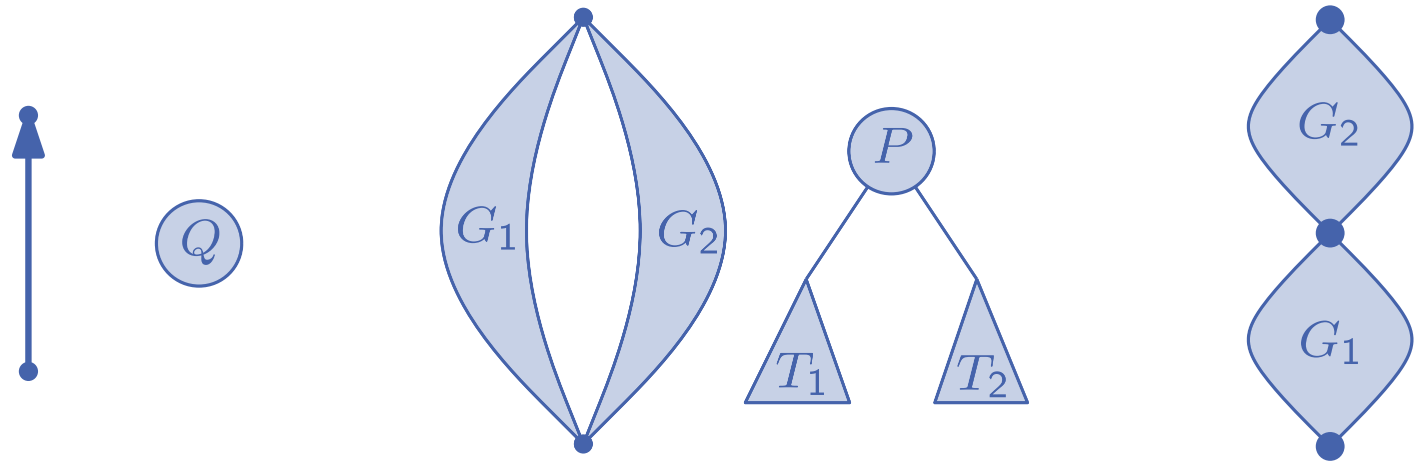
Karlsruhe Institute of Technology

Lemma

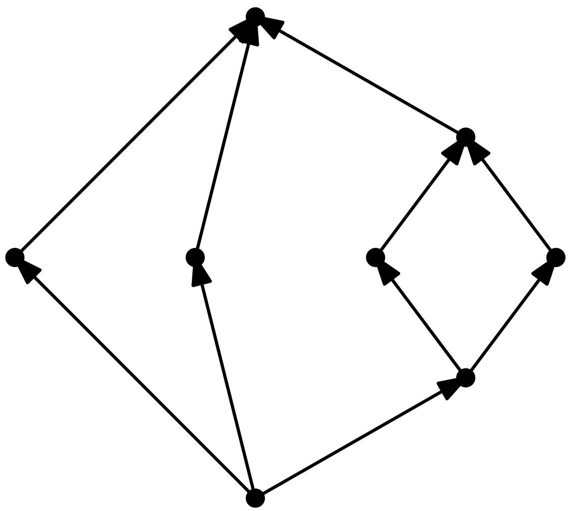
Series-parallel graphs are acyclic and planar.

In order to proof this statement we can use a **decomposition tree** of G, which is a binary tree T with nodes of three types: S,P and Q-type.

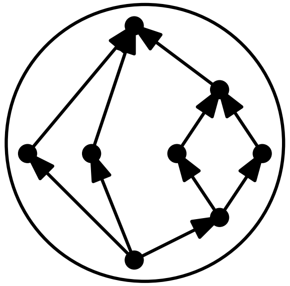
- If G is a single edge, then the corresponding node is Q-node
- If G is a parallel composition of G_1 (with tree T_1) and G_2 (with tree T_2), then the root of T is P-node and T_1 is its left subtree, T_2 is its right subtree
- If G is a series composition of G_1 (with tree T_1) and G_2 (with tree T_2), then the root of T is S-node and T_1 is its left subtree, T_2 is its right subtree



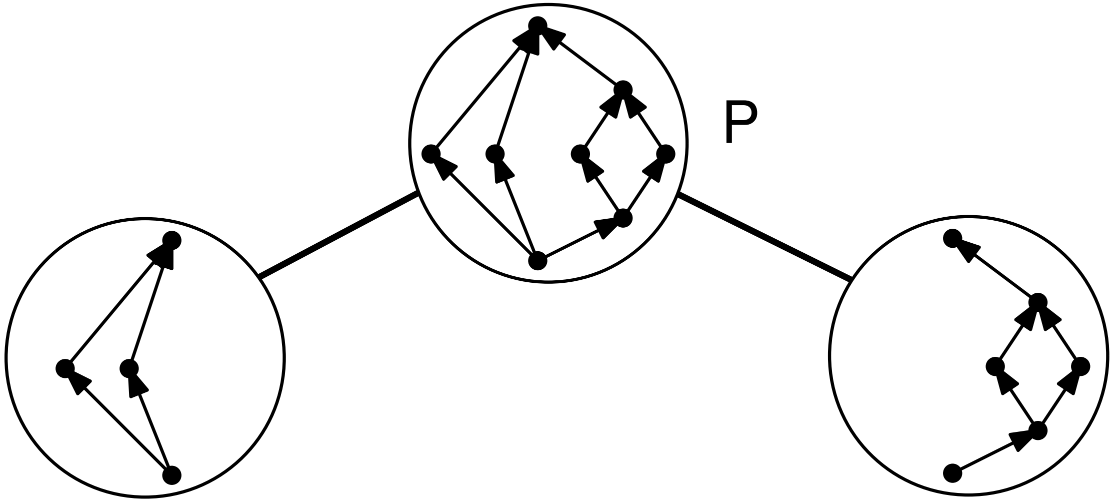
Series-parallel Graphs. Decomposition Example.



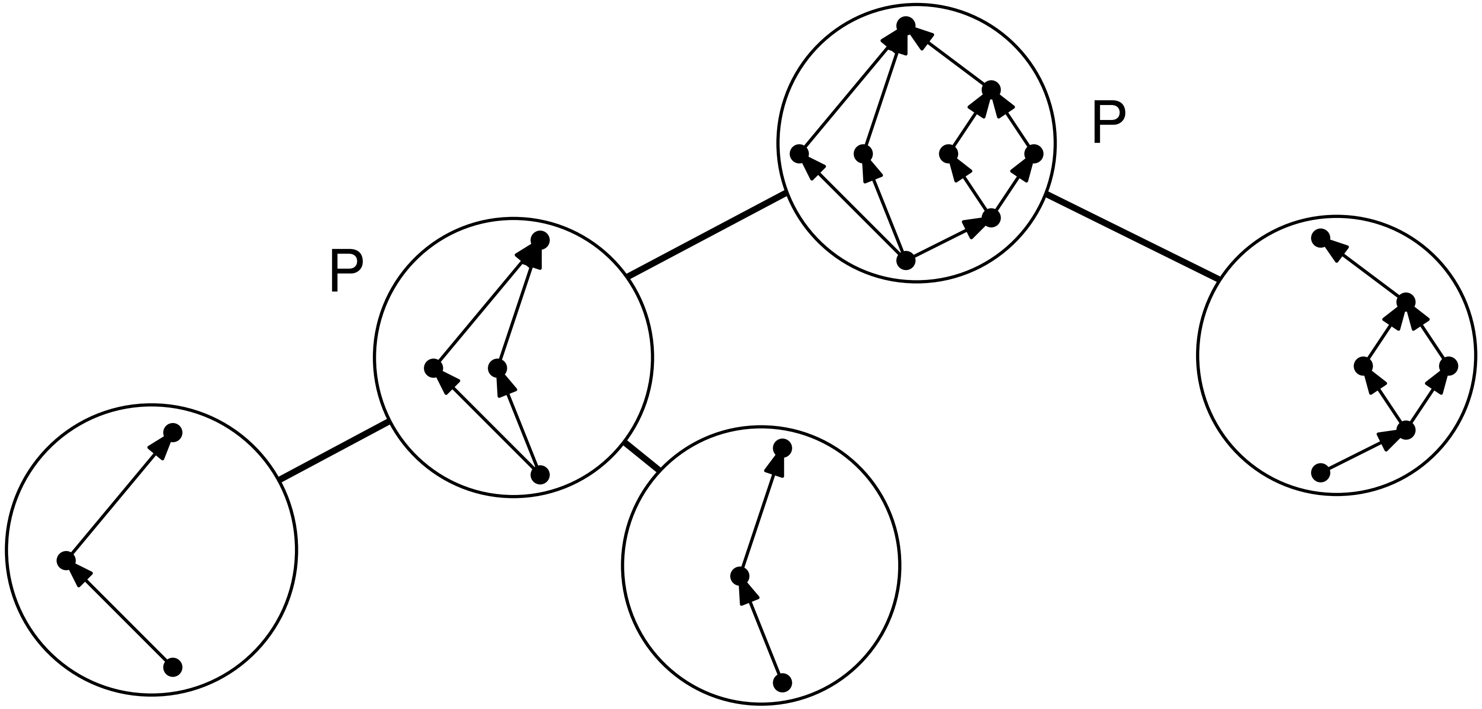
Series-parallel Graphs. Decomposition Example.



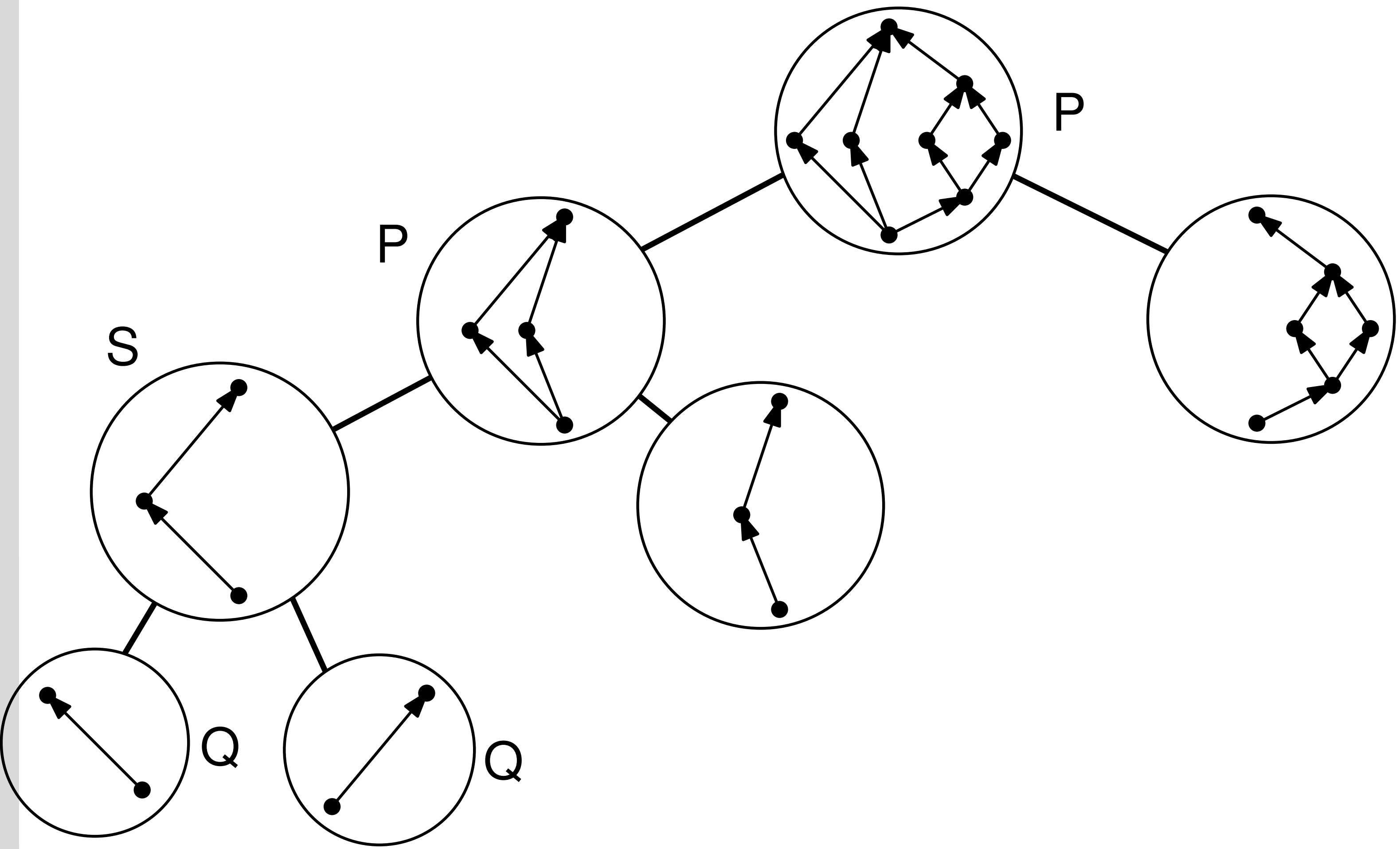
Series-parallel Graphs. Decomposition Example.



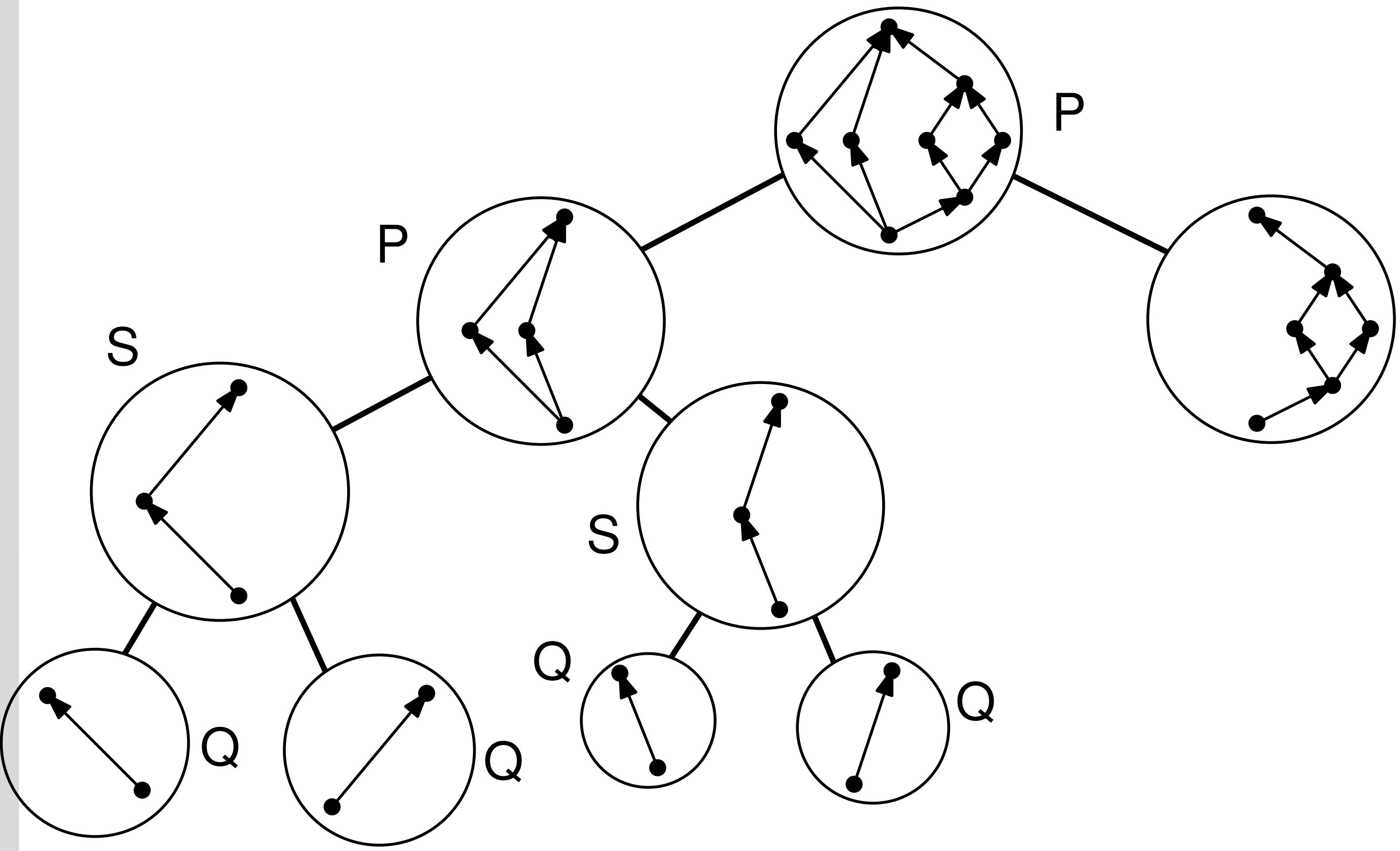
Series-parallel Graphs. Decomposition Example.



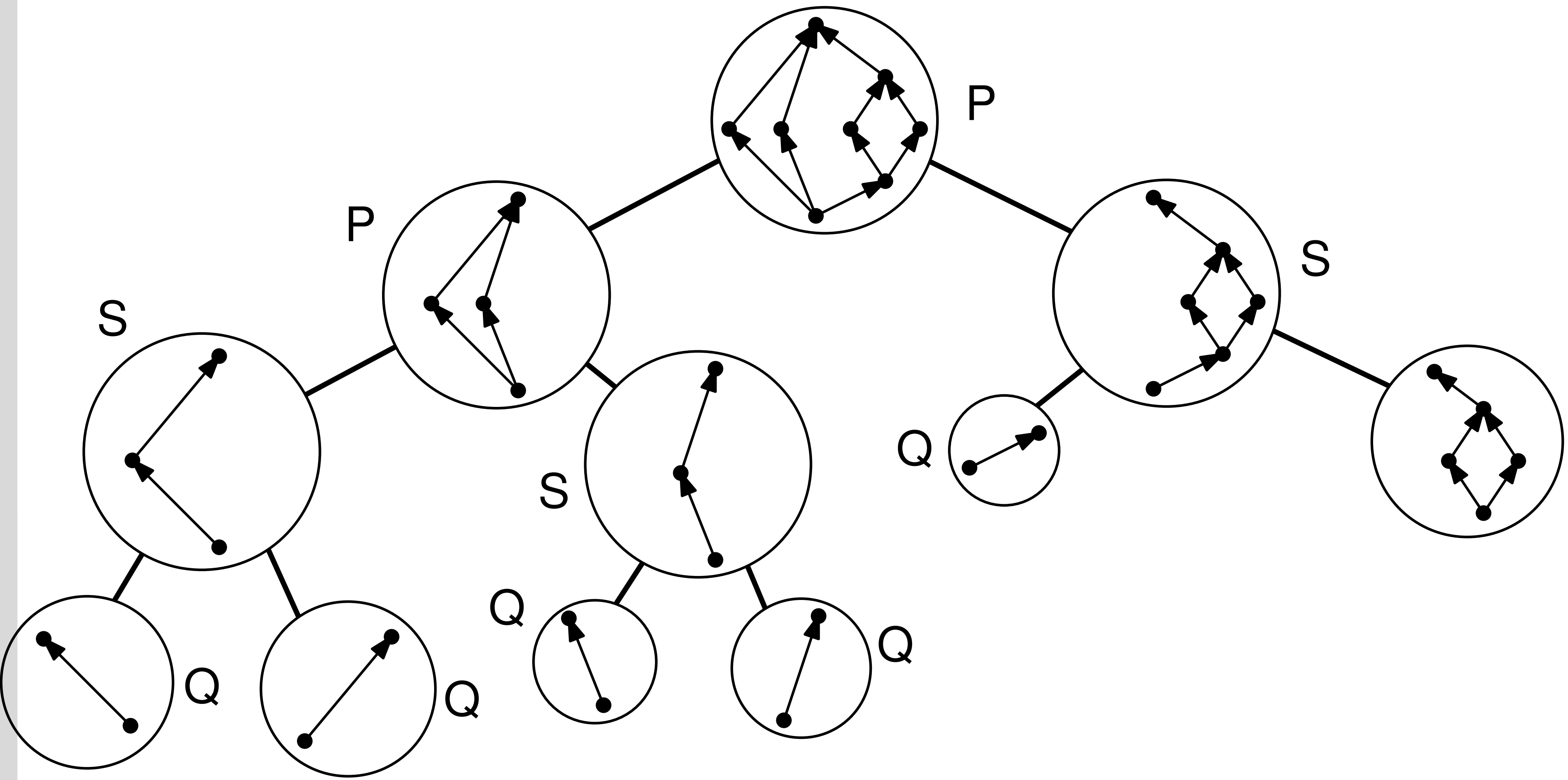
Series-parallel Graphs. Decomposition Example.



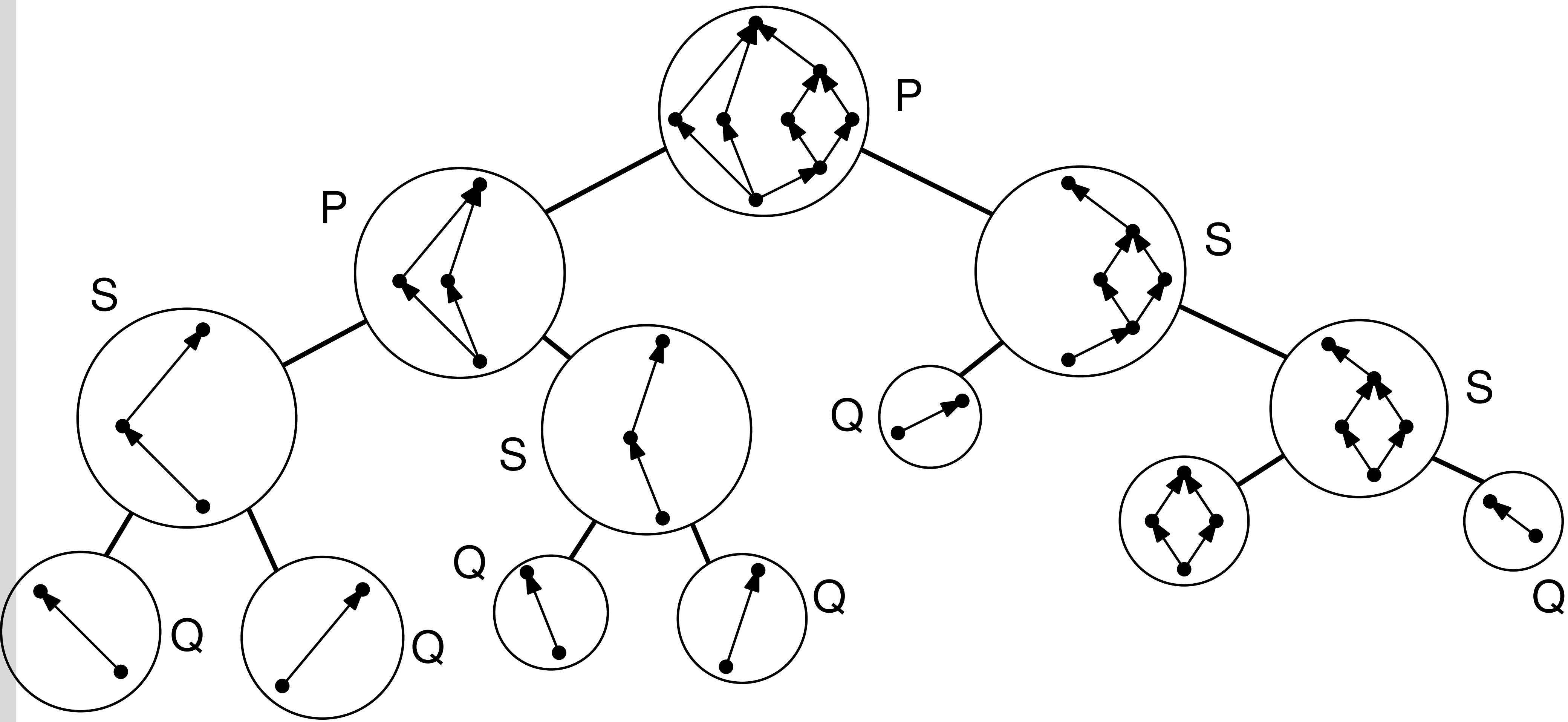
Series-parallel Graphs. Decomposition Example



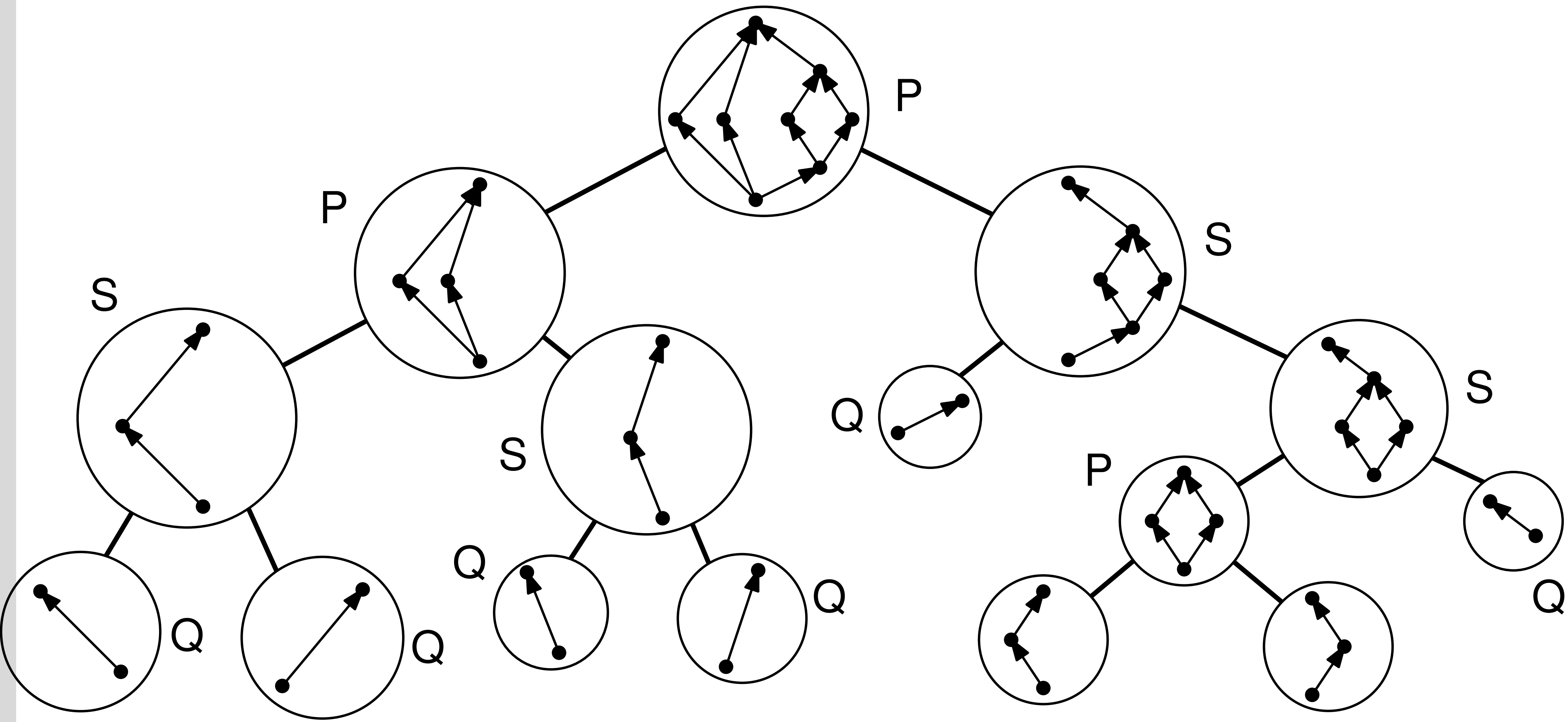
Series-parallel Graphs. Decomposition Example



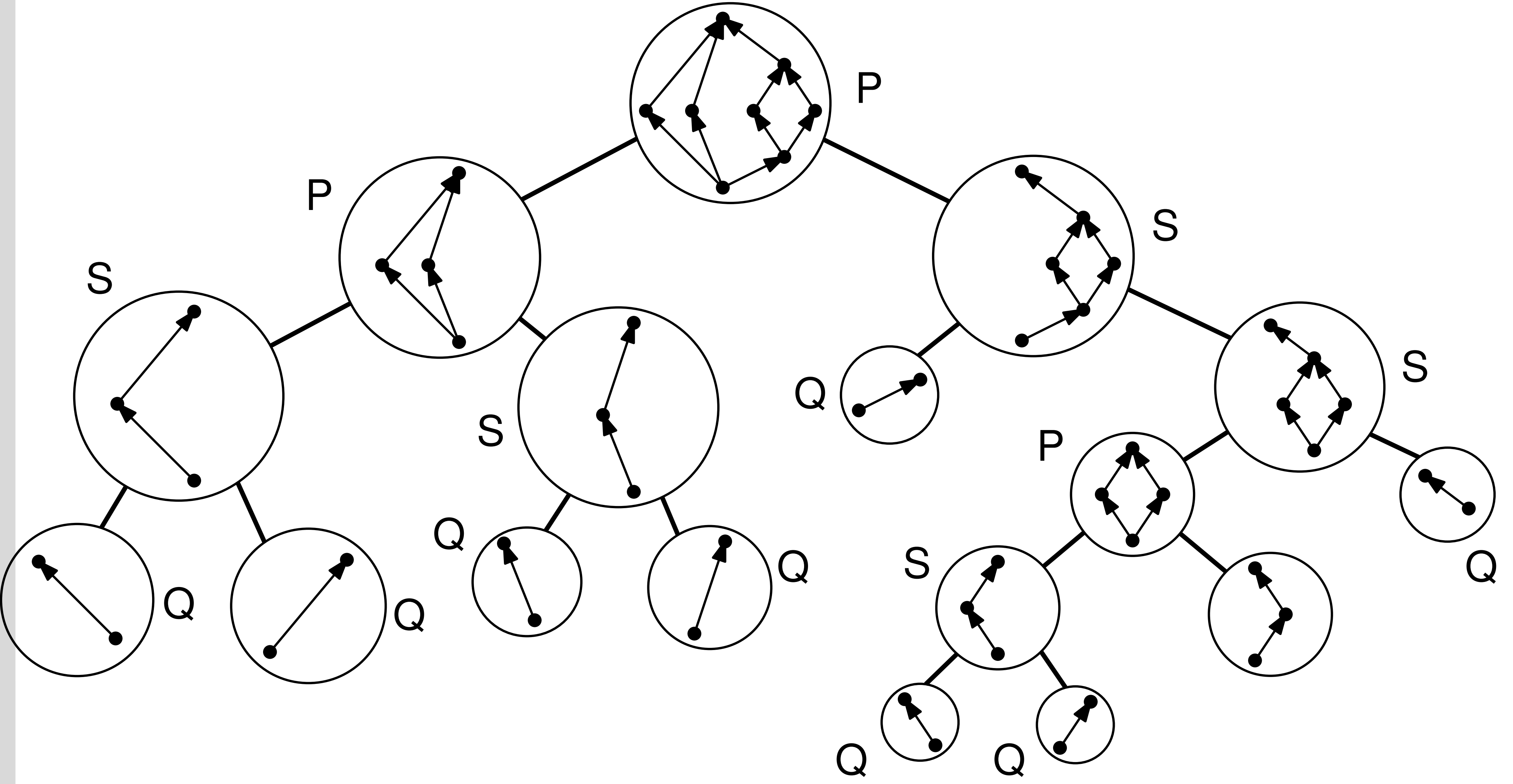
Series-parallel Graphs. Decomposition Example.



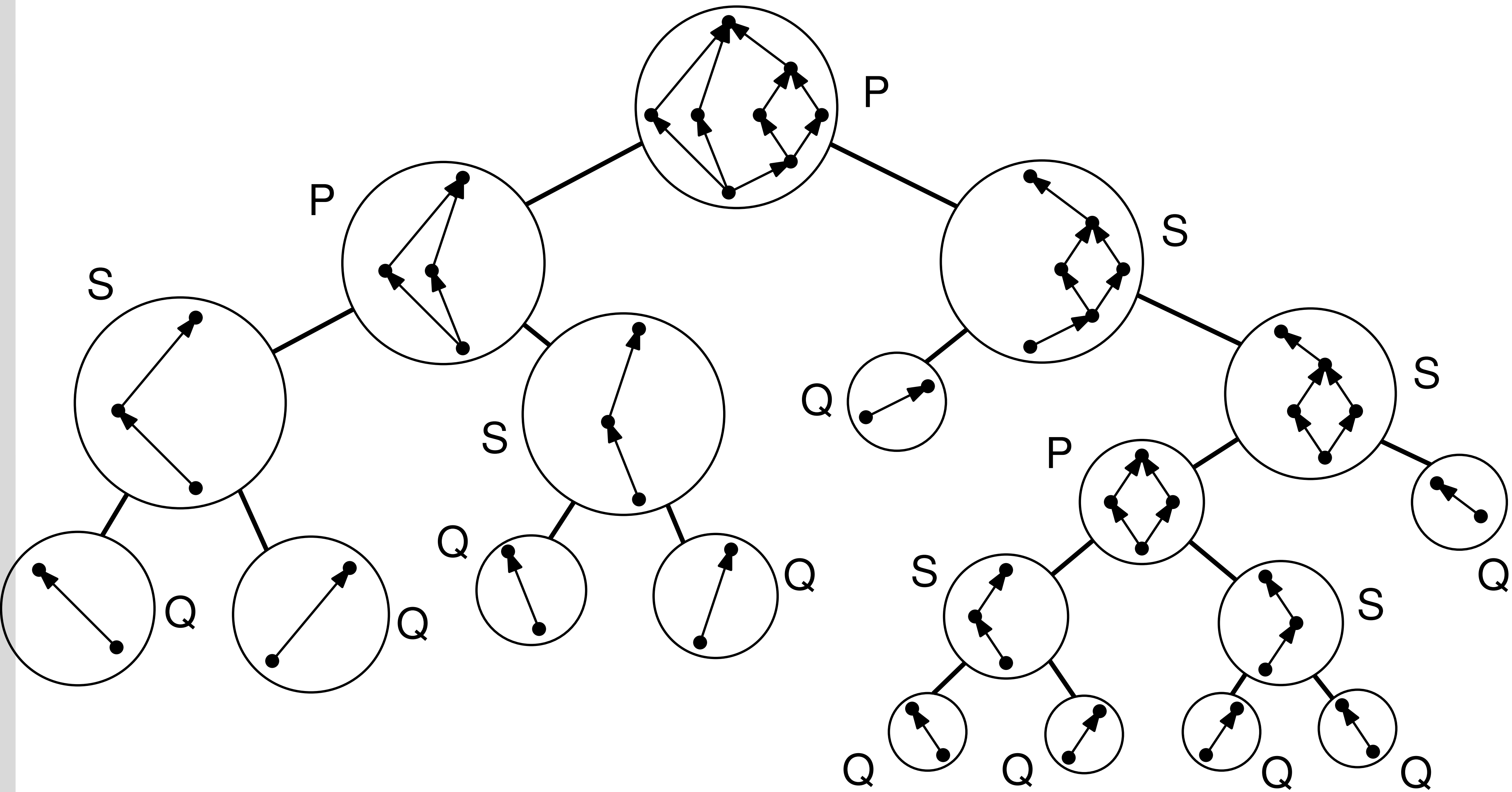
Series-parallel Graphs. Decomposition Example



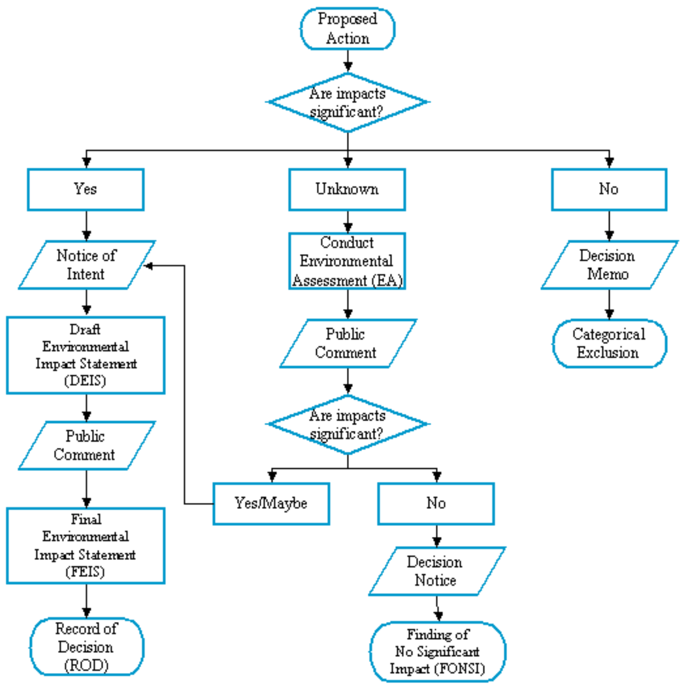
Series-parallel Graphs. Decomposition Example.



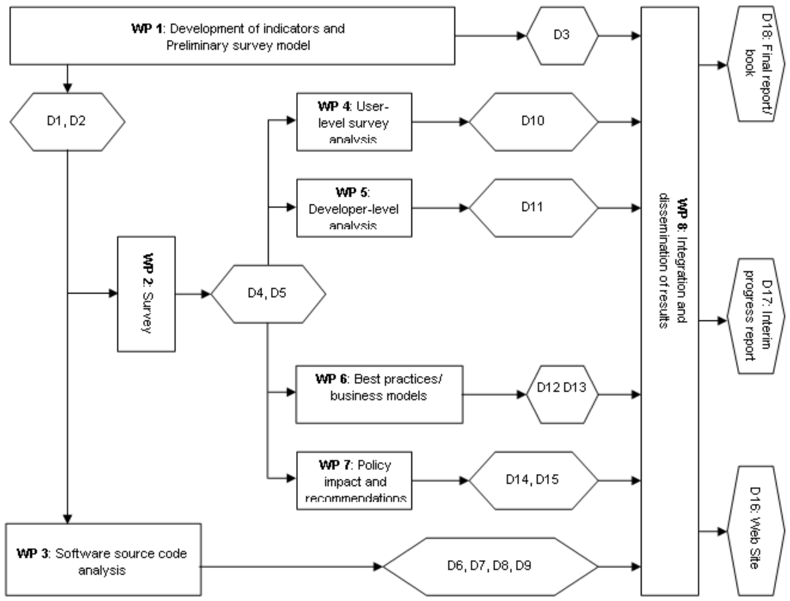
Series-parallel Graphs. Decomposition Example.



Series-parallel Graphs. Applications.



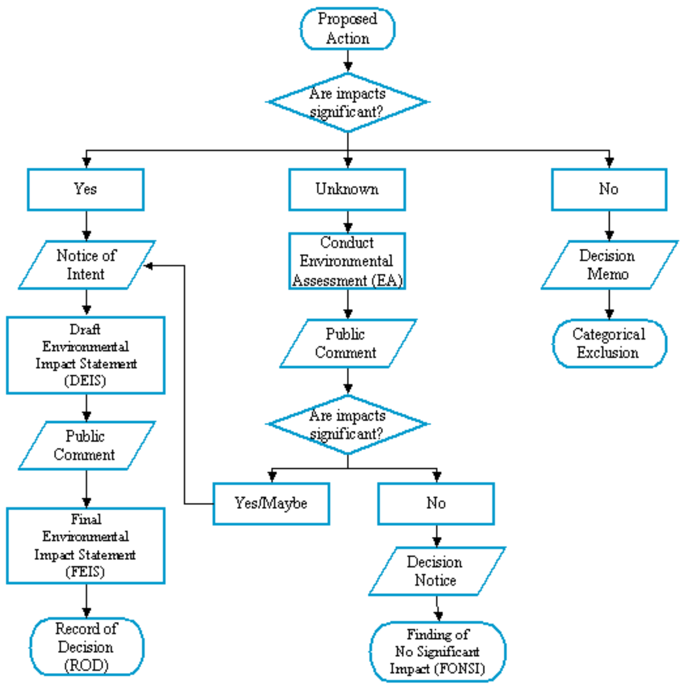
Flowcharts



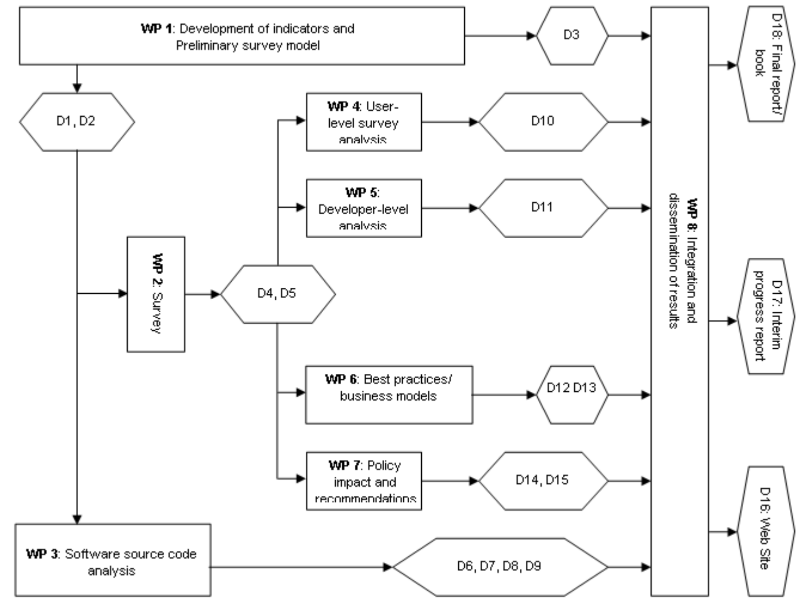
PERT-Diagrams

(Program Evaluation and Review Technique)

Series-parallel Graphs. Applications.



Flowcharts



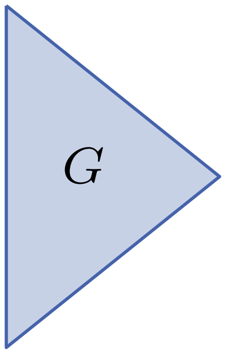
PERT-Diagrams

(Program Evaluation and Review Technique)

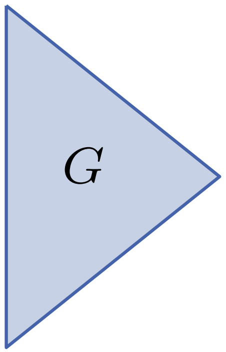
Computational Complexity: Linear time algorithms for $\mathcal{NP}$-hard problems (e.g. Maximum Matching, Maximum Independent Set, Hamiltonian Completion)

Straight-line Drawing of SP-Graphs

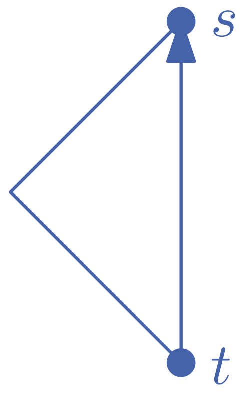
- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$



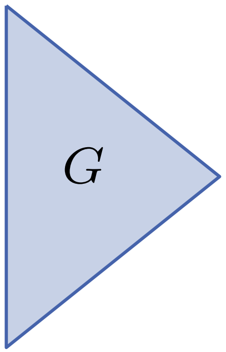
Straight-line Drawing of SP-Graphs



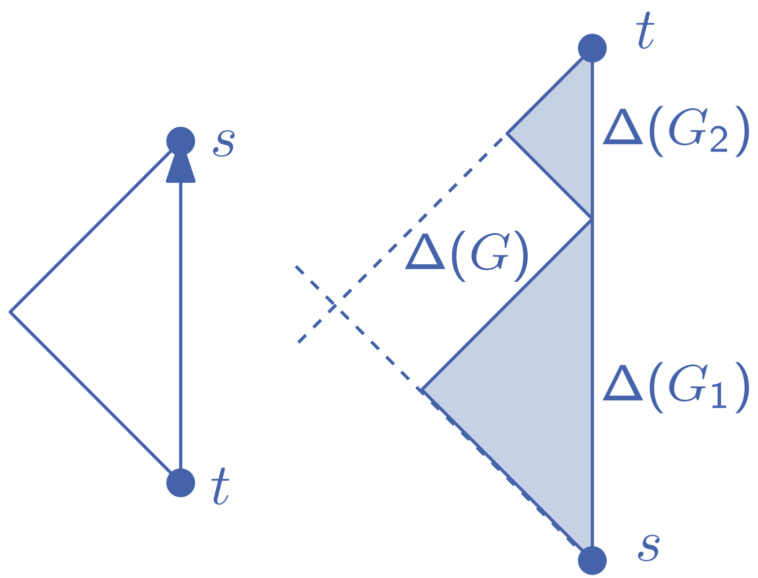
- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):



Straight-line Drawing of SP-Graphs

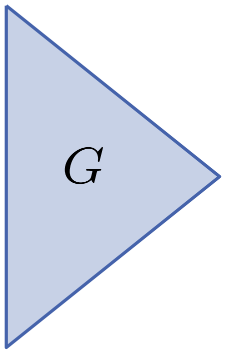


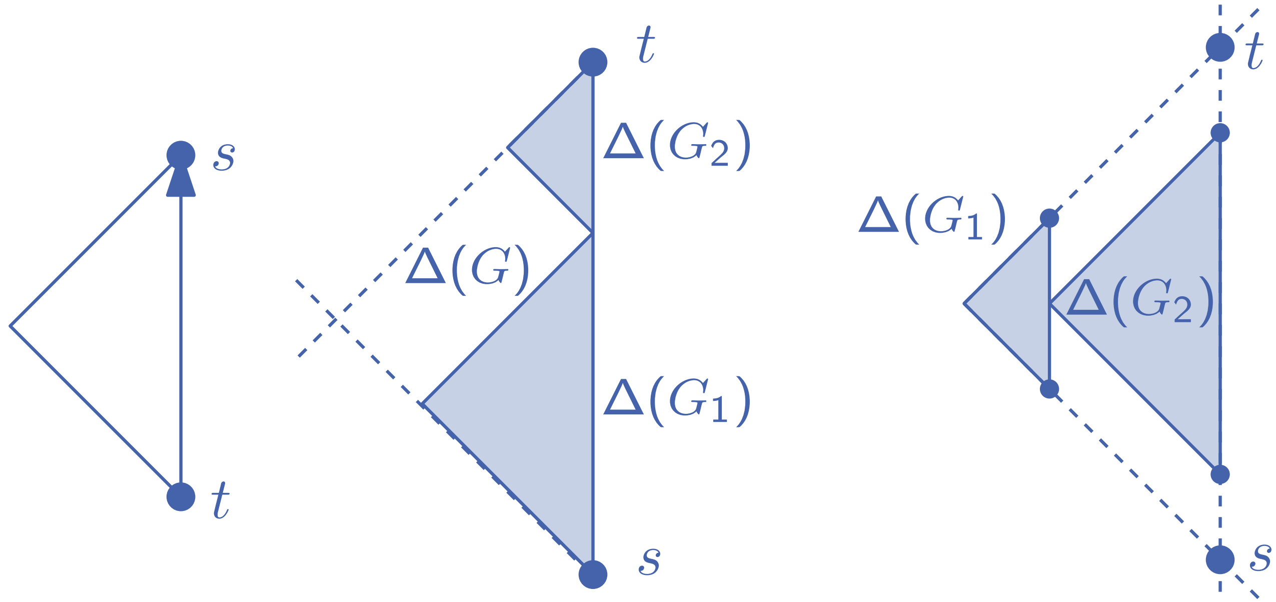
- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)



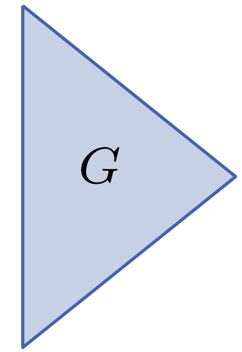
Straight-line Drawing of SP-Graphs

- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)

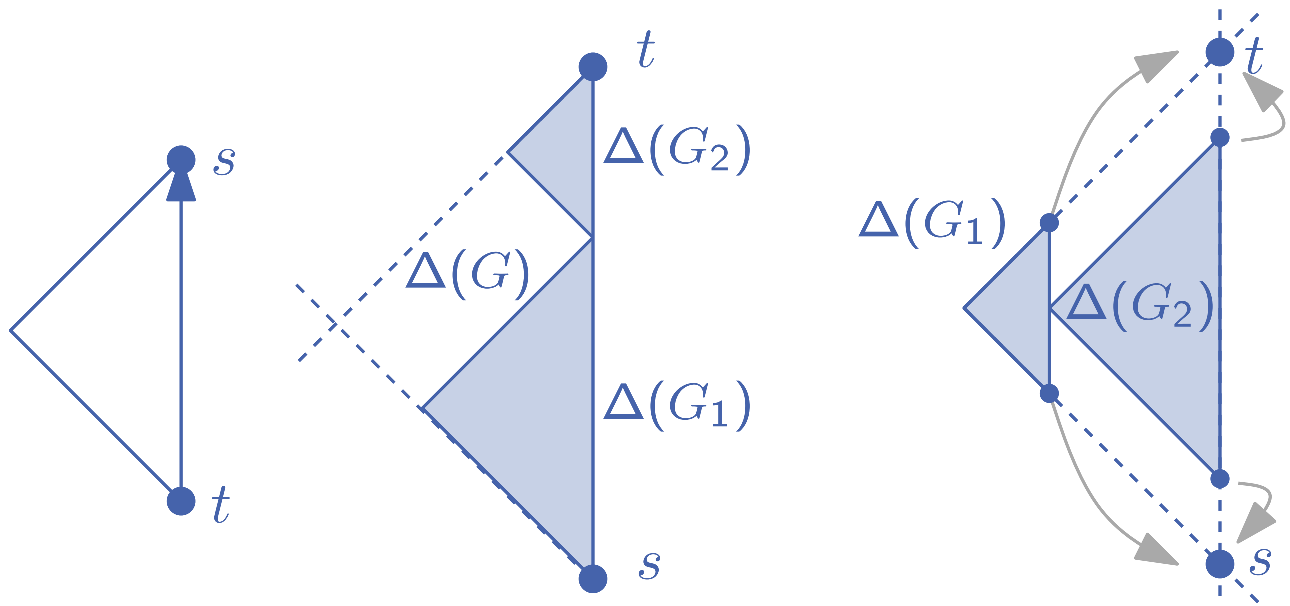




Straight-line Drawing of SP-Graphs

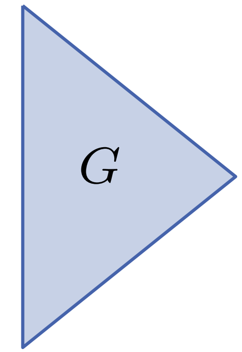


- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)

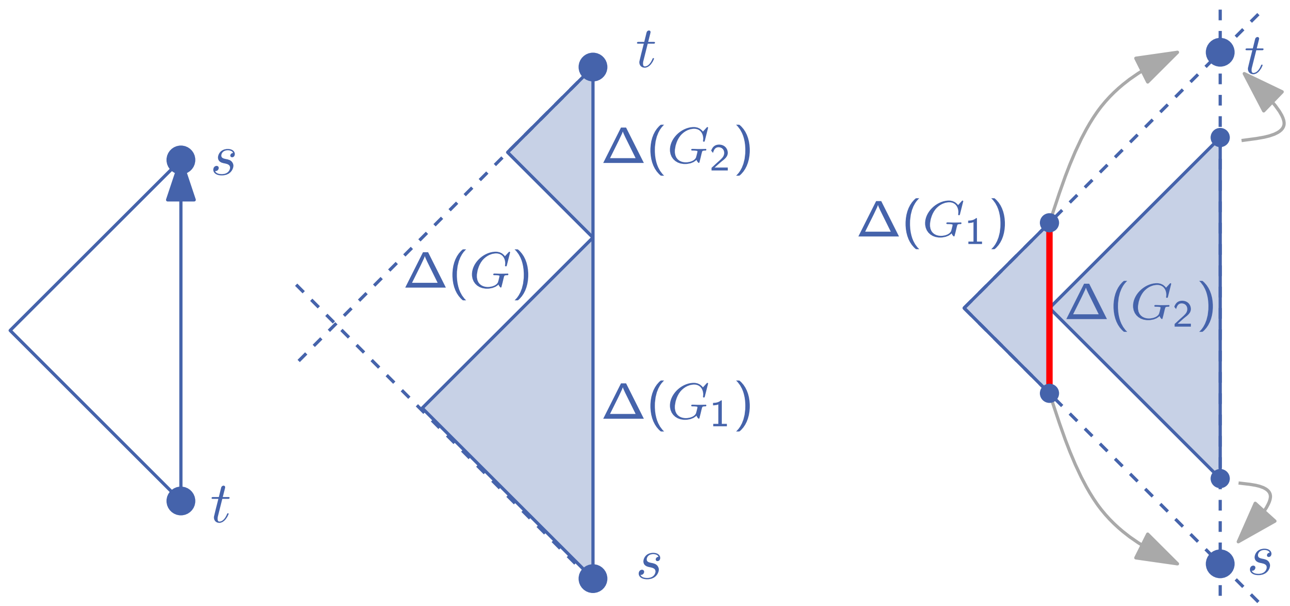


Institut für Theoretische Informatik
Lehrstuhl Algorithmik I

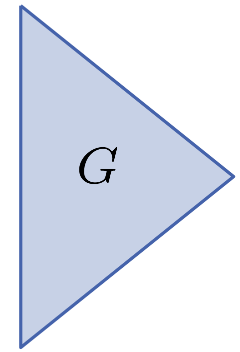
Straight-line Drawing of SP-Graphs



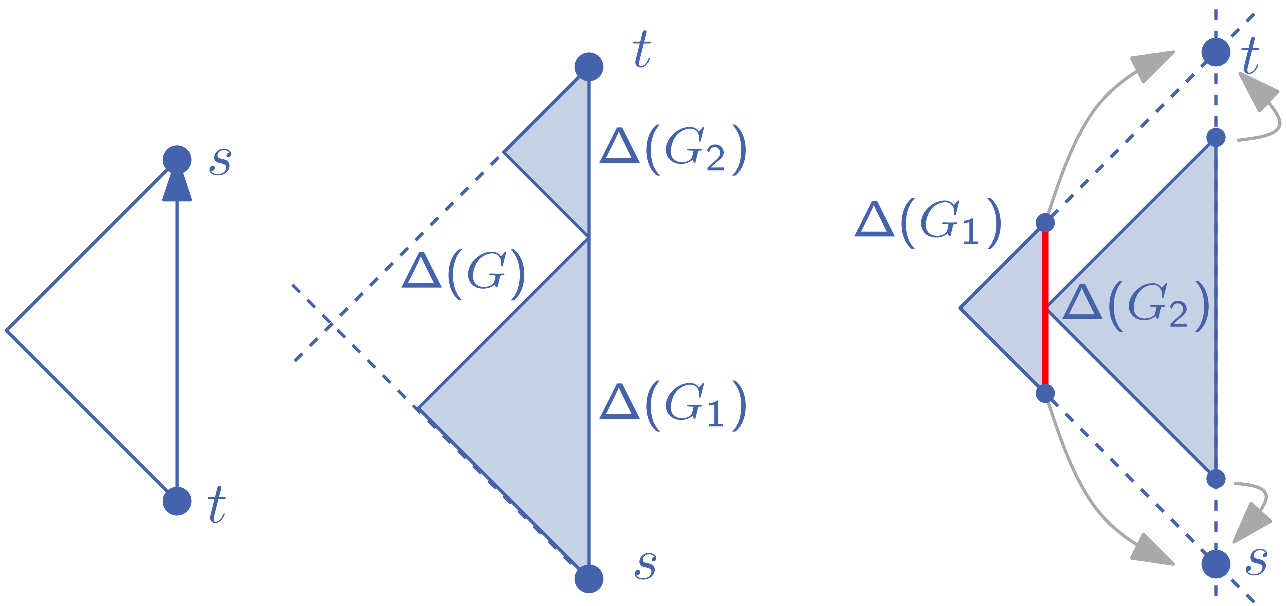
- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)



Straight-line Drawing of SP-Graphs

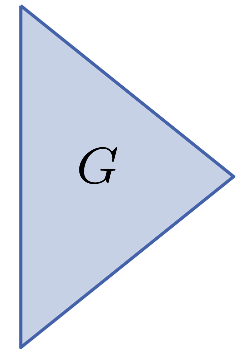


- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)

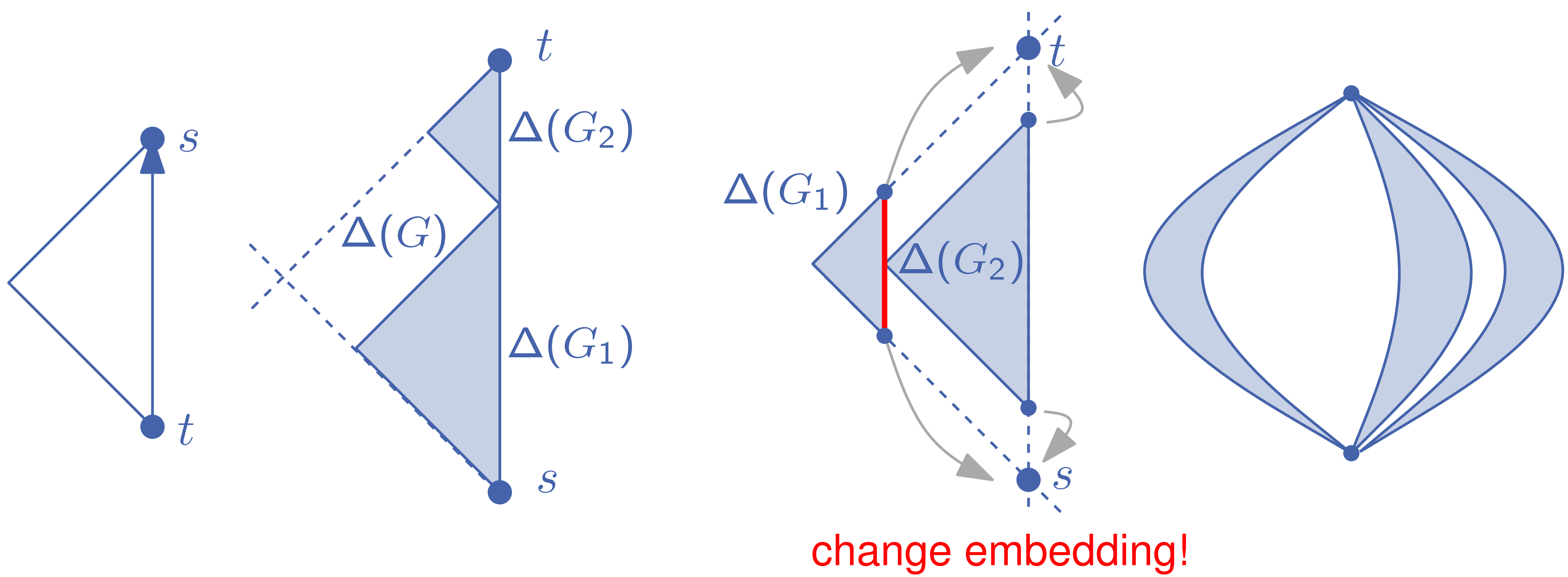


change embedding!

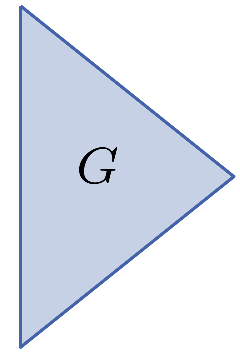
Straight-line Drawing of SP-Graphs



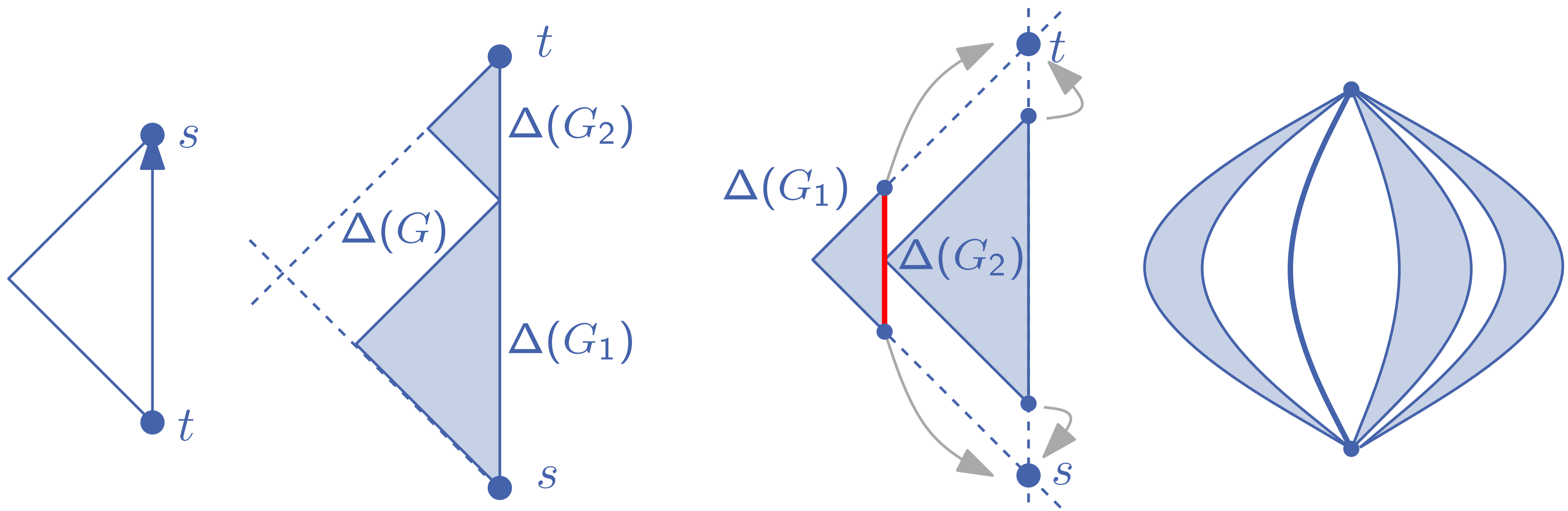
- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)



Straight-line Drawing of SP-Graphs



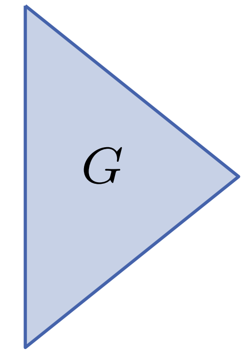
- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)

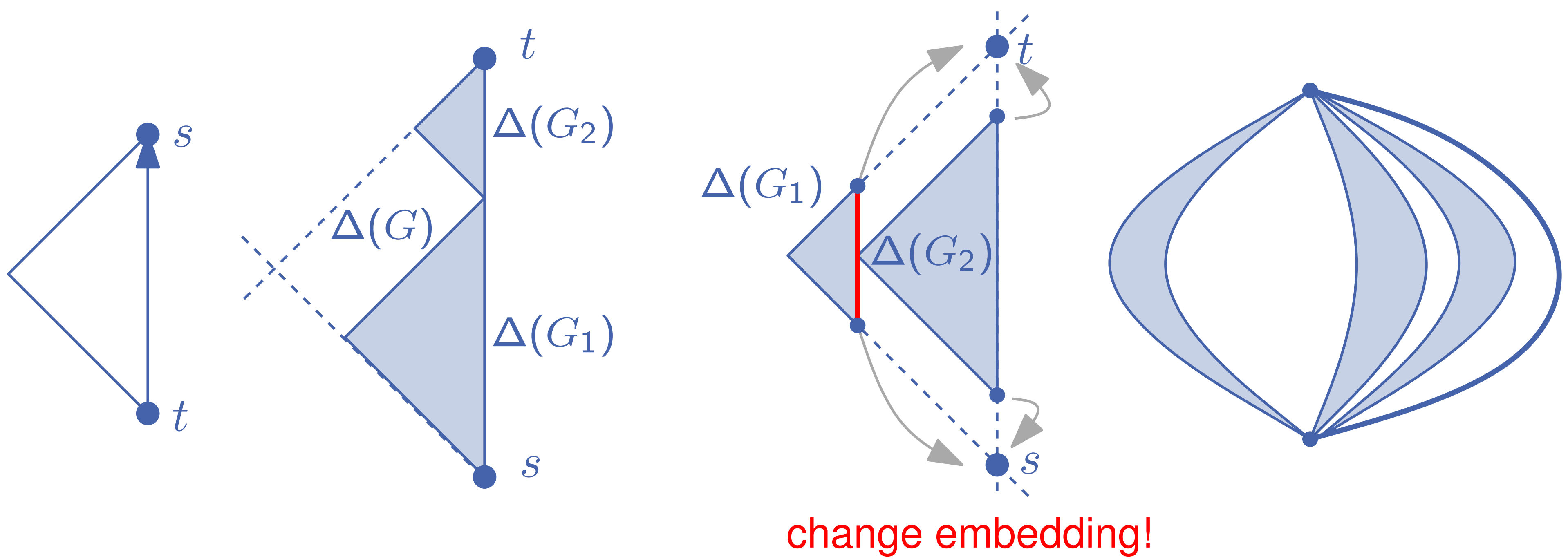


change embedding!

Straight-line Drawing of SP-Graphs

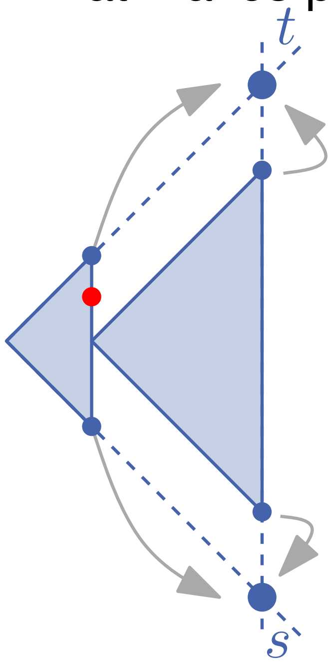
- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)





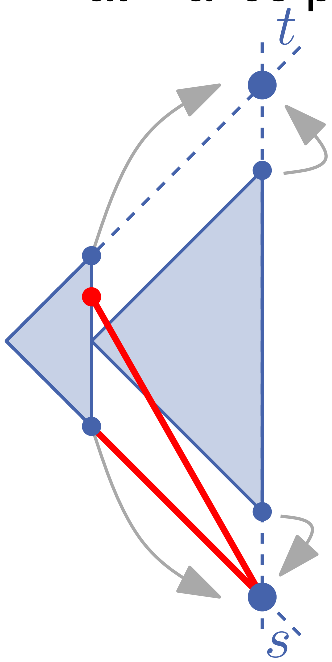
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



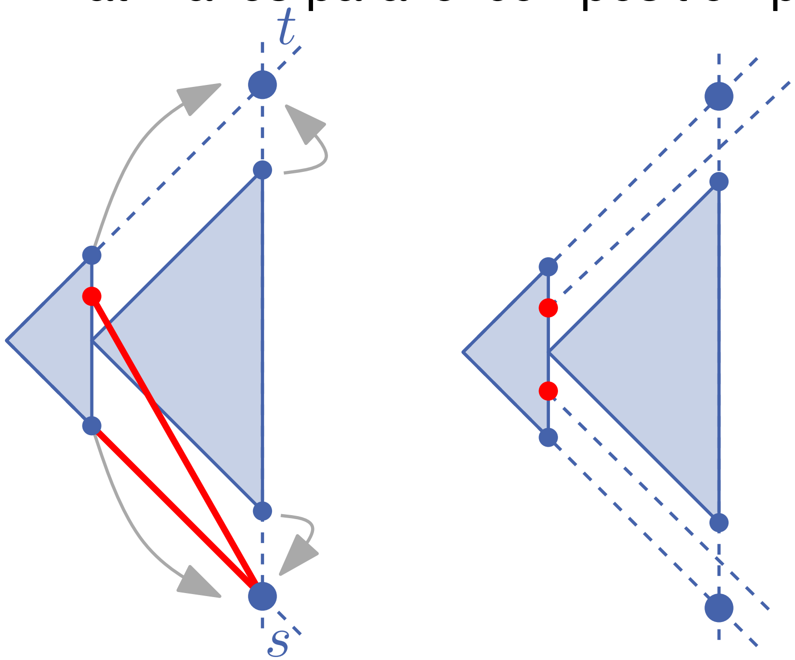
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



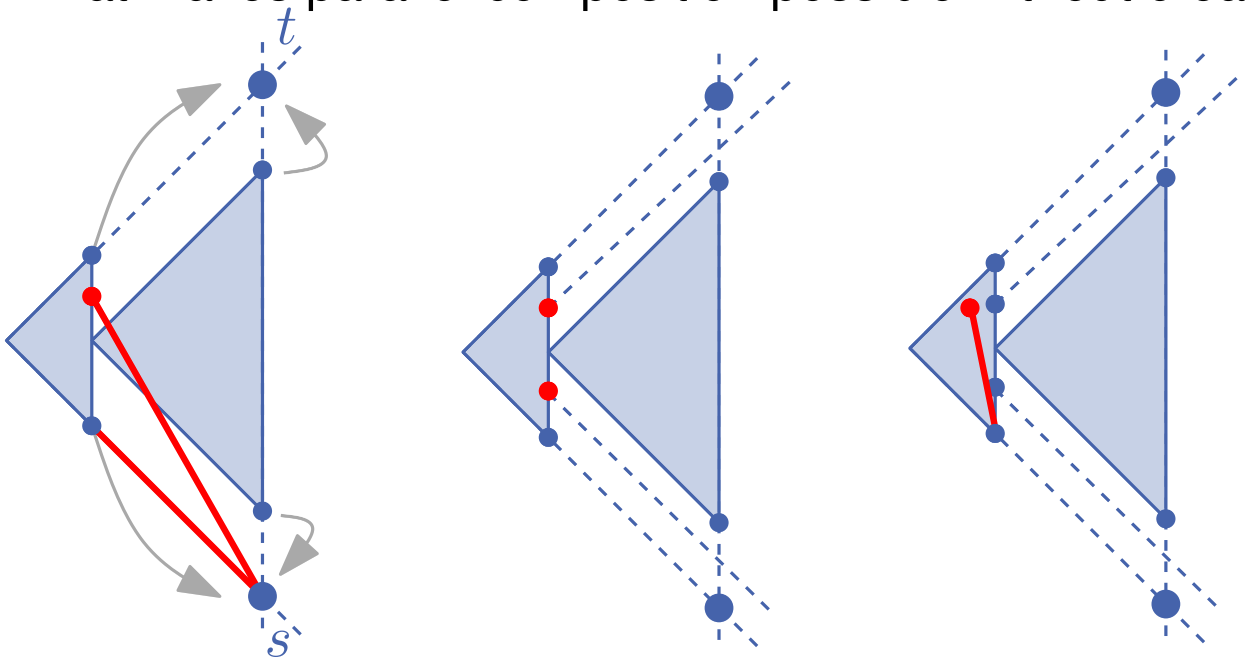
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



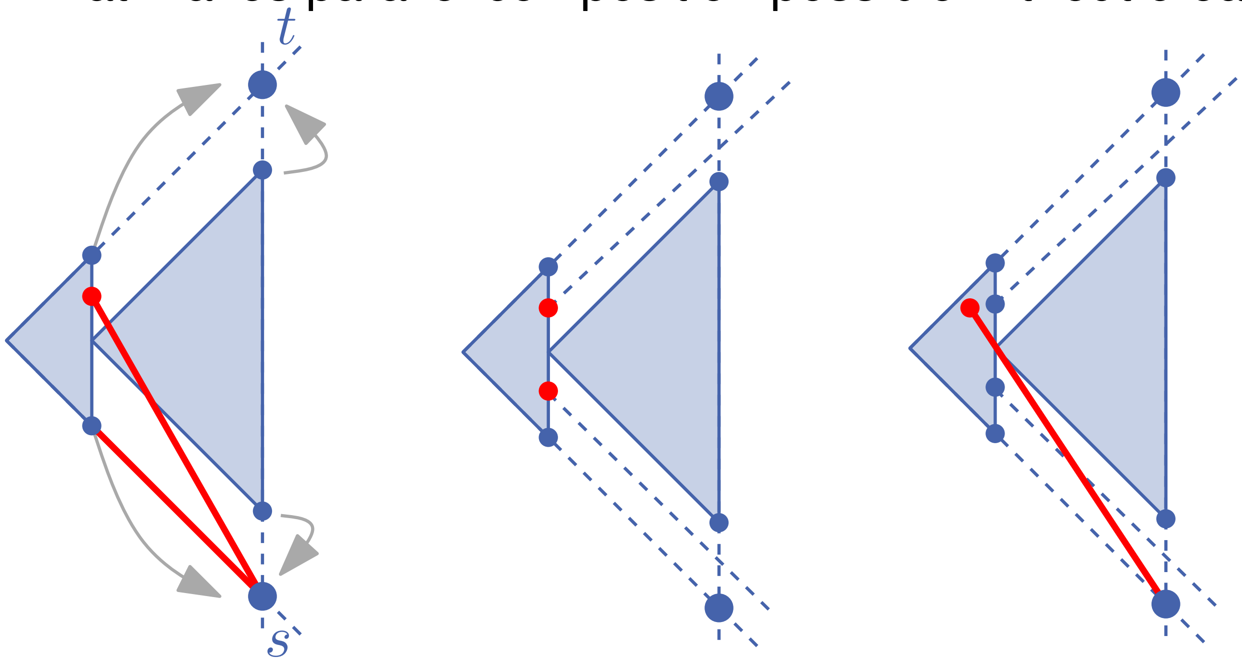
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



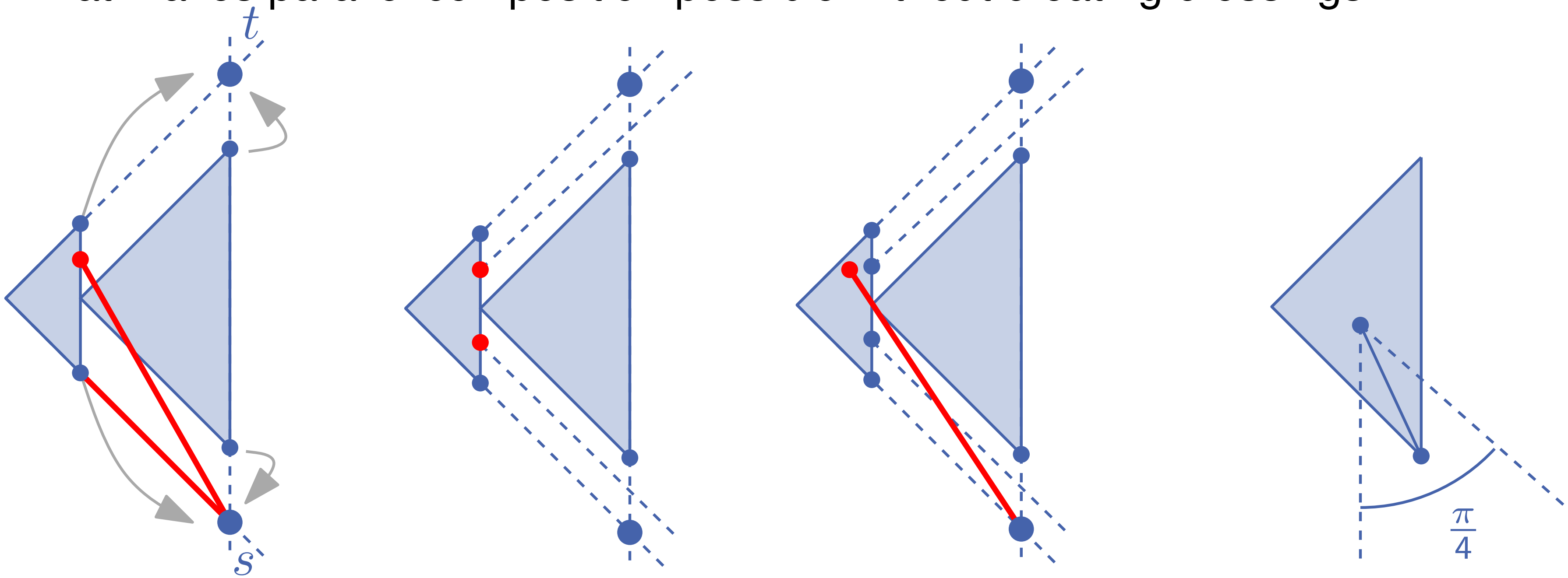
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



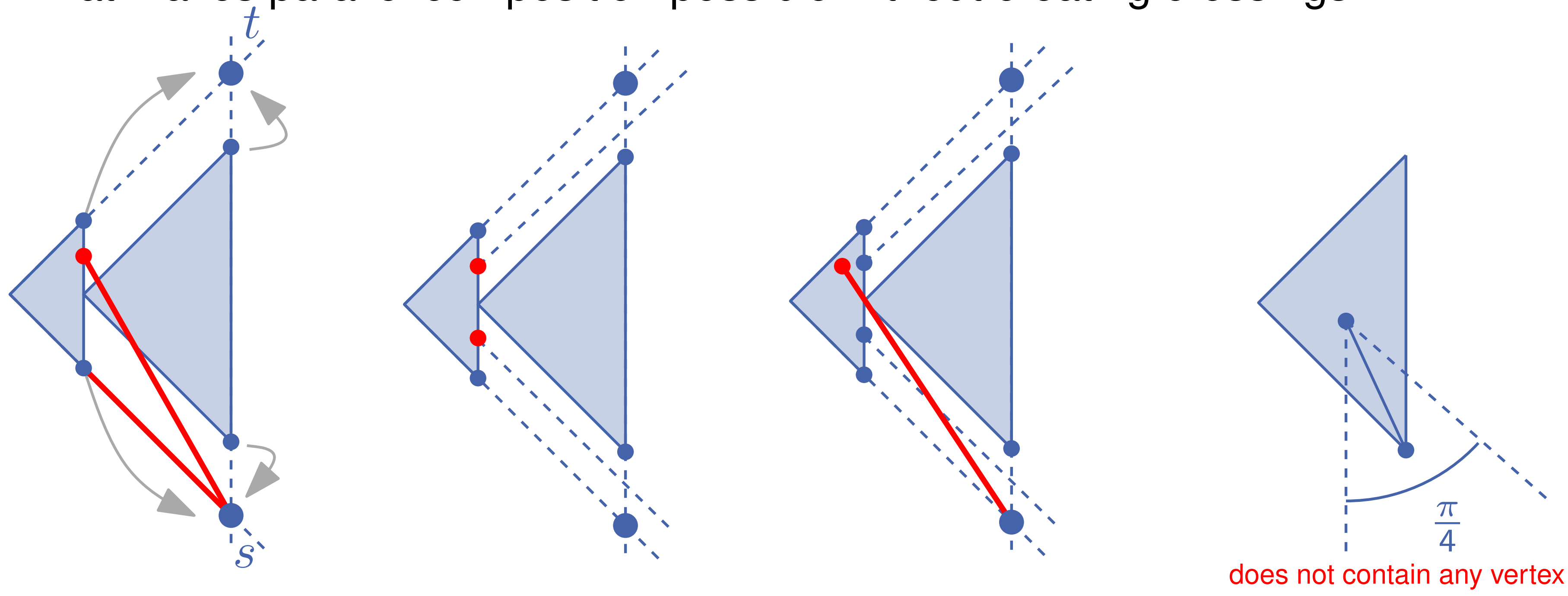
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



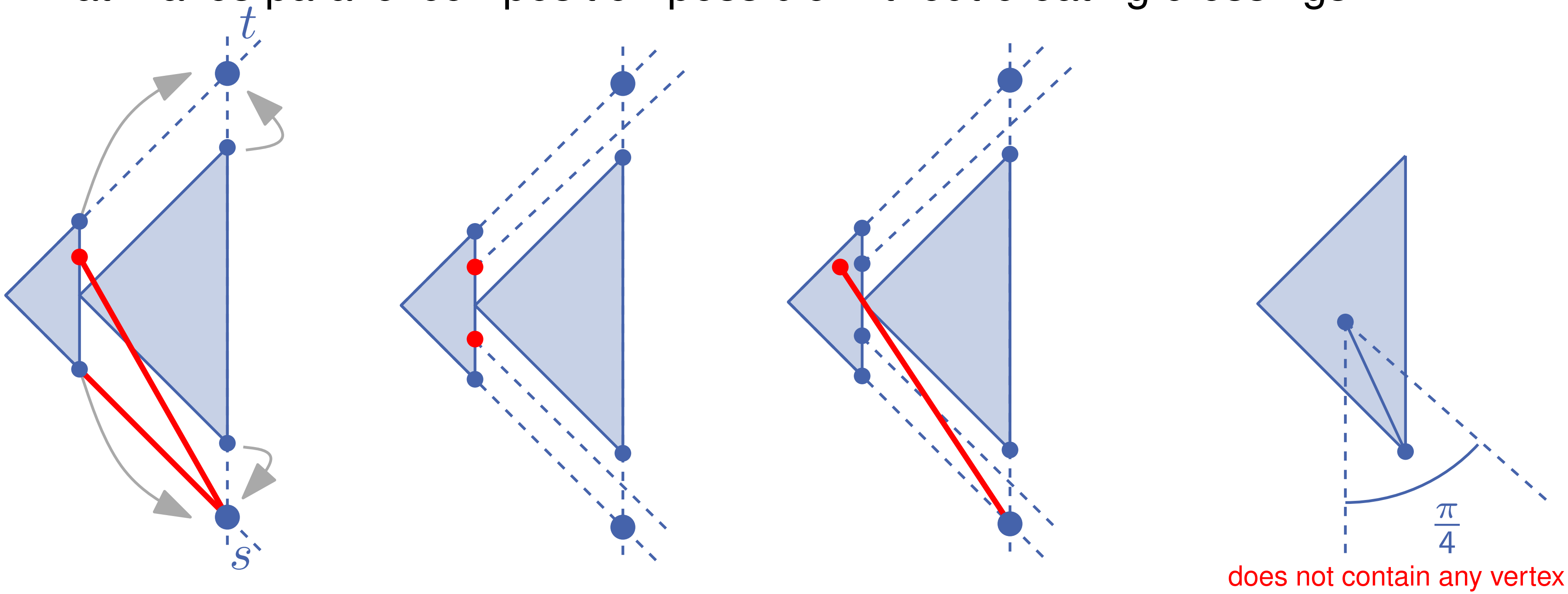
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



Straight-line Drawing of SP-Graphs

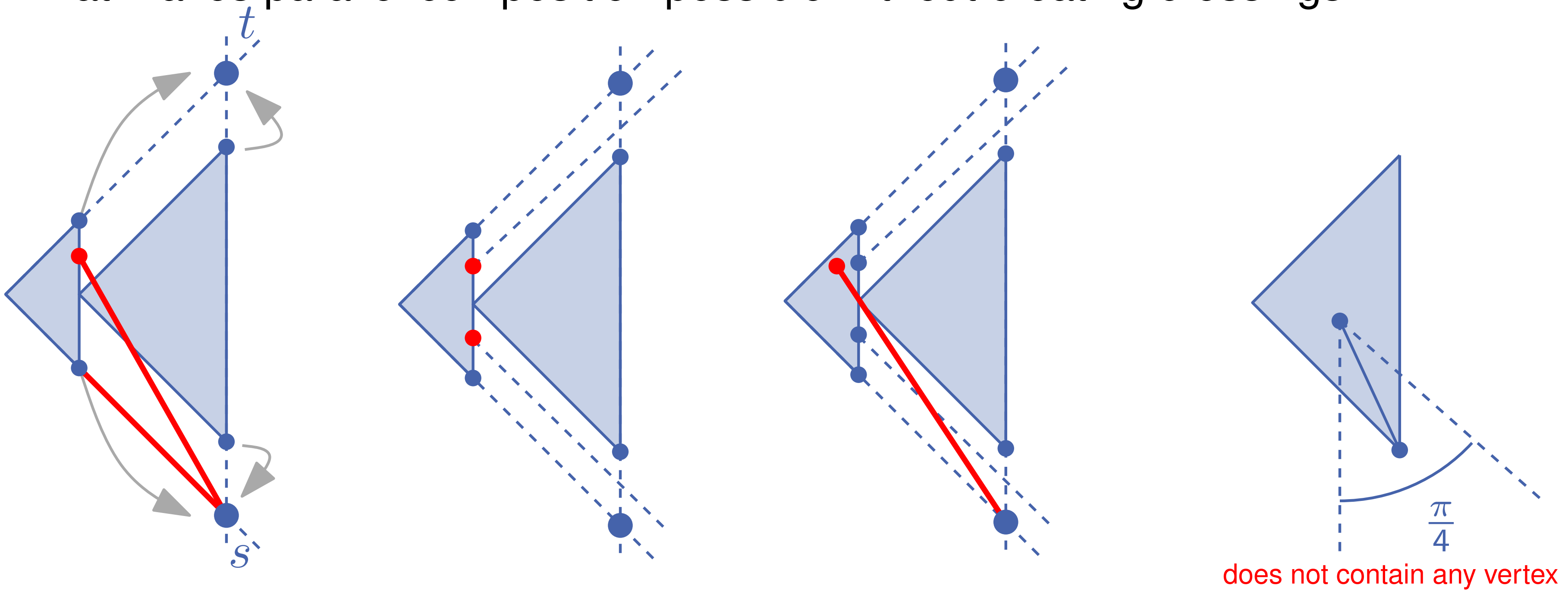
- What makes parallel composition possible without creating crossings?



- This condition can be preserved during the induction step.

Straight-line Drawing of SP-Graphs

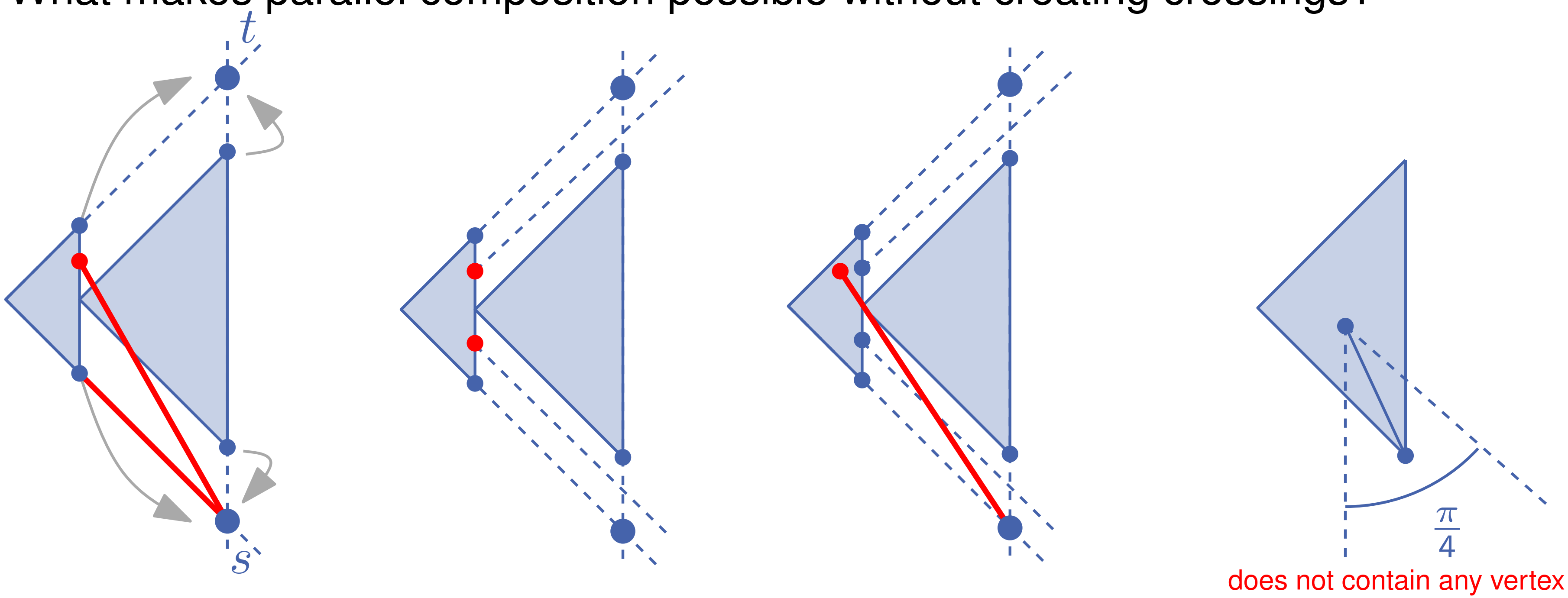
- What makes parallel composition possible without creating crossings?



- This condition can be preserved during the induction step.
- The area of the drawing is?

Straight-line Drawing of SP-Graphs

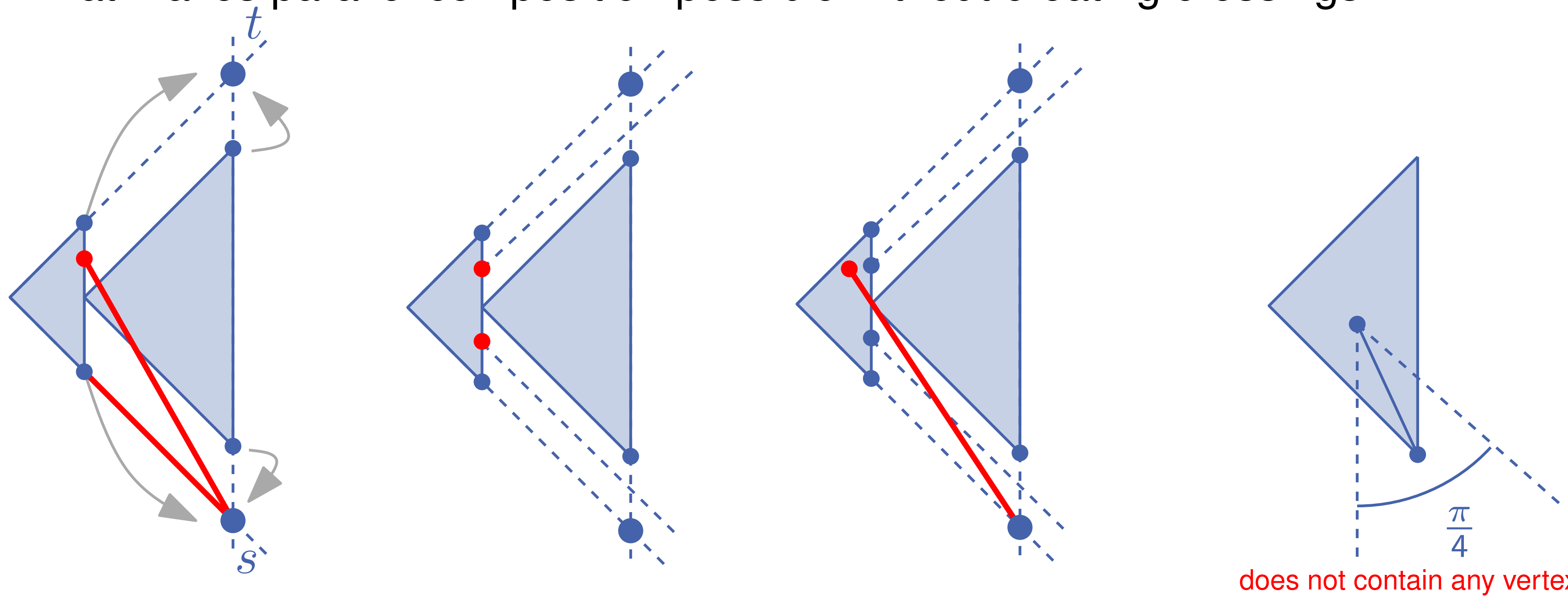
- What makes parallel composition possible without creating crossings?



- This condition can be preserved during the induction step.
- The area of the drawing is? $O(m^2), m$ is the number of edges

Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



- This condition can be preserved during the induction step.
- The area of the drawing is? $O(m^2), m$ is the number of edges

Theorem

A series-parallel graph G (**with variable embedding**) admits an **upward** straigh-line drawing with $O(n^2)$ area. The isomorphic componencts of G have congruent drawings up to a translation.

Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

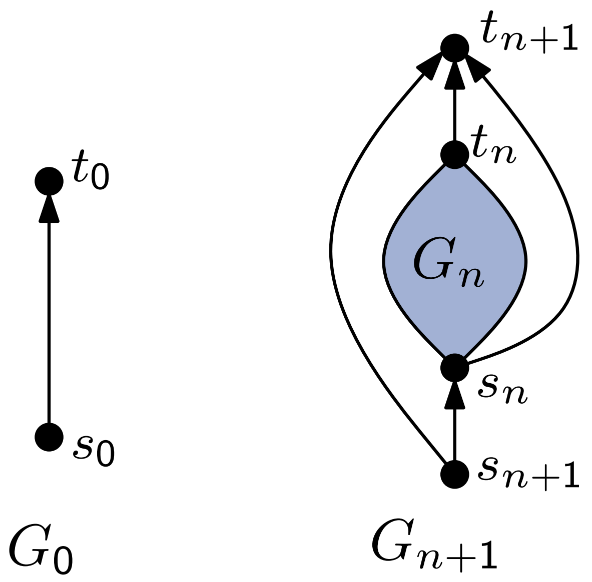
There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:

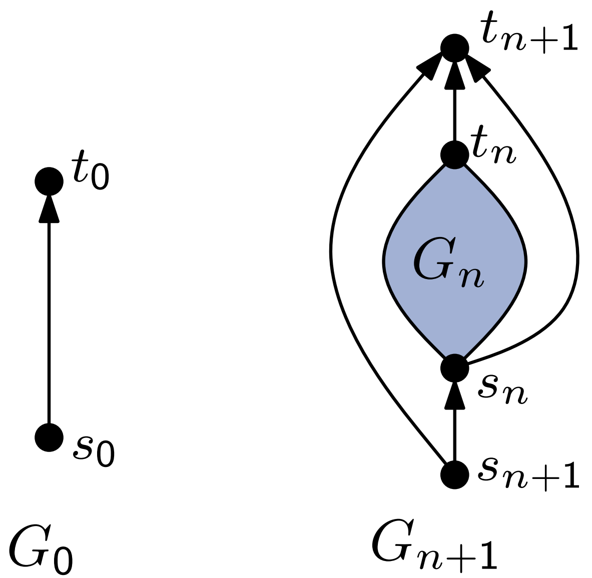


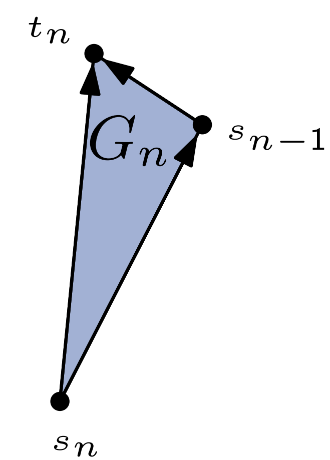
Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:



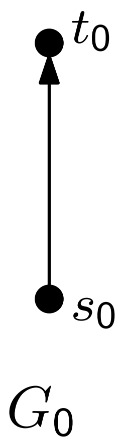


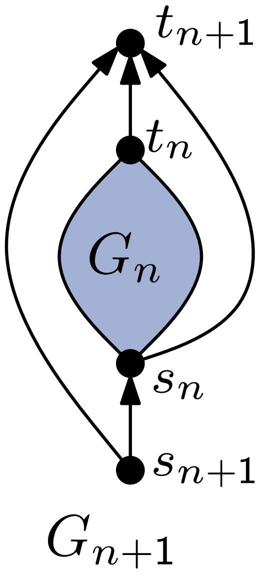
Lower Bound for the Area

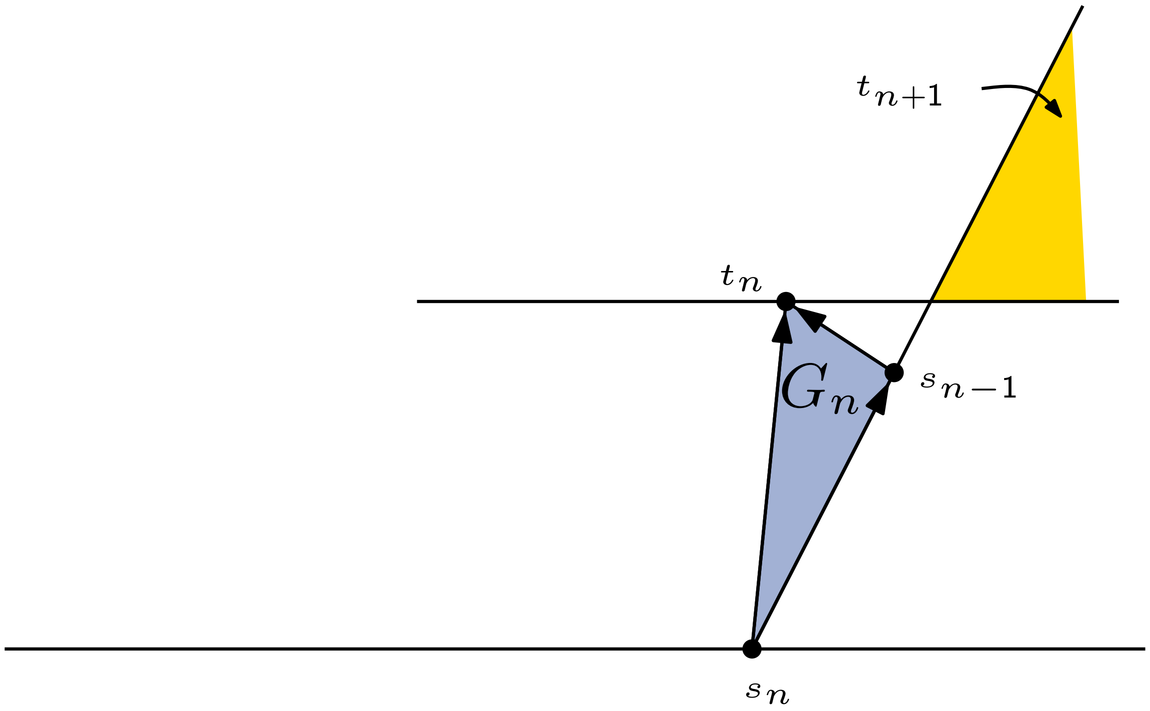
Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:





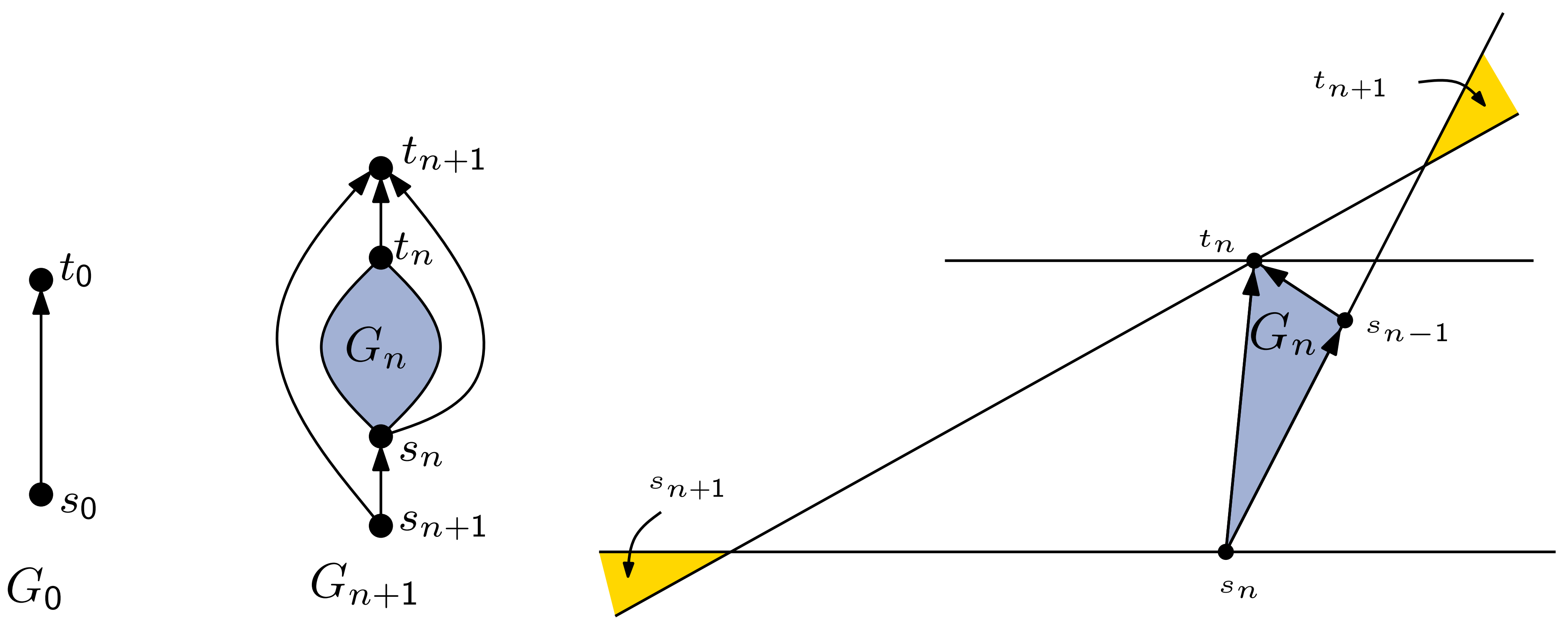


Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:

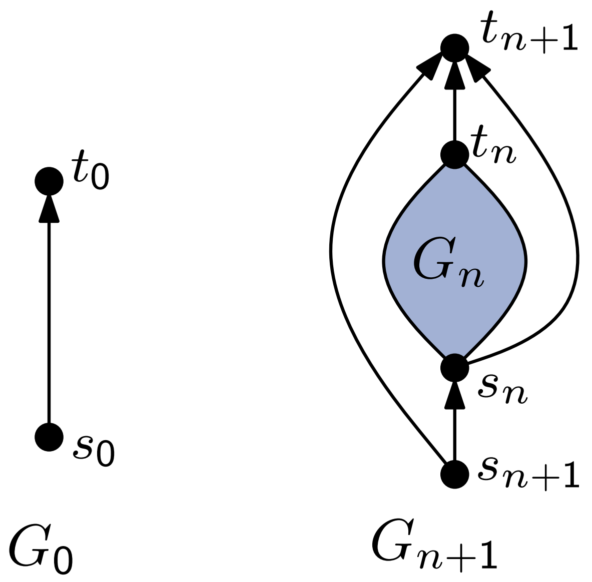


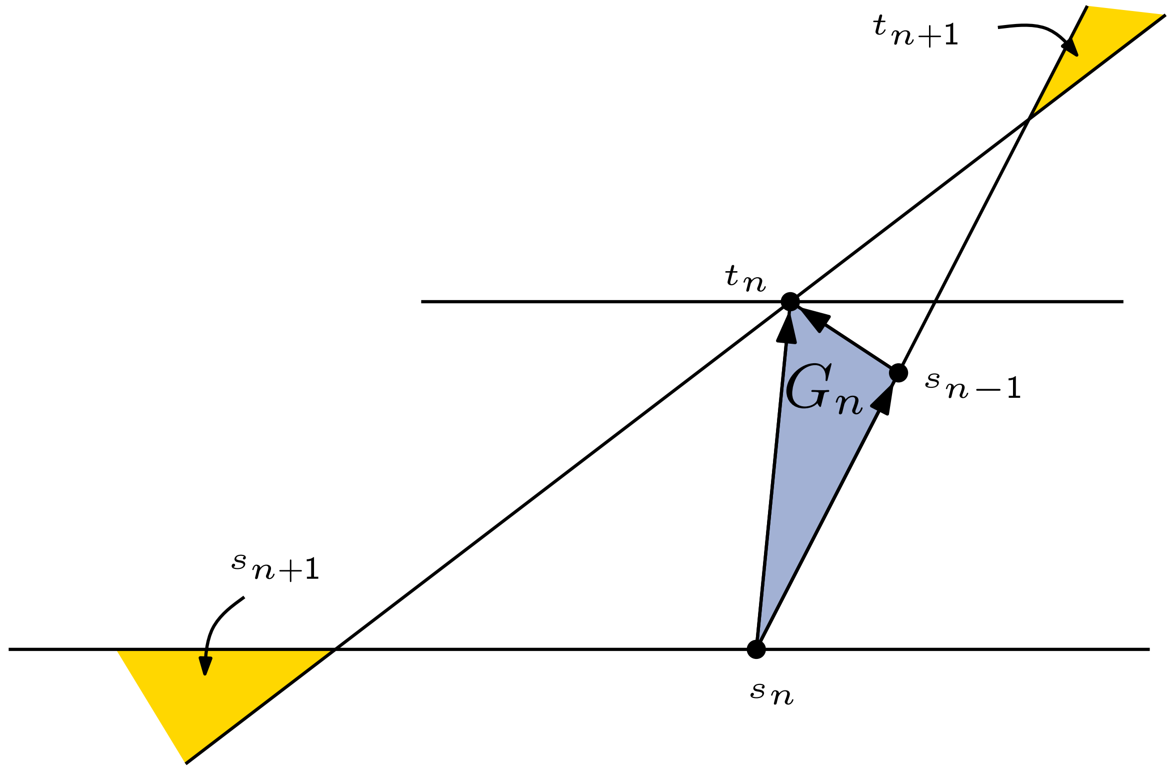
Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:



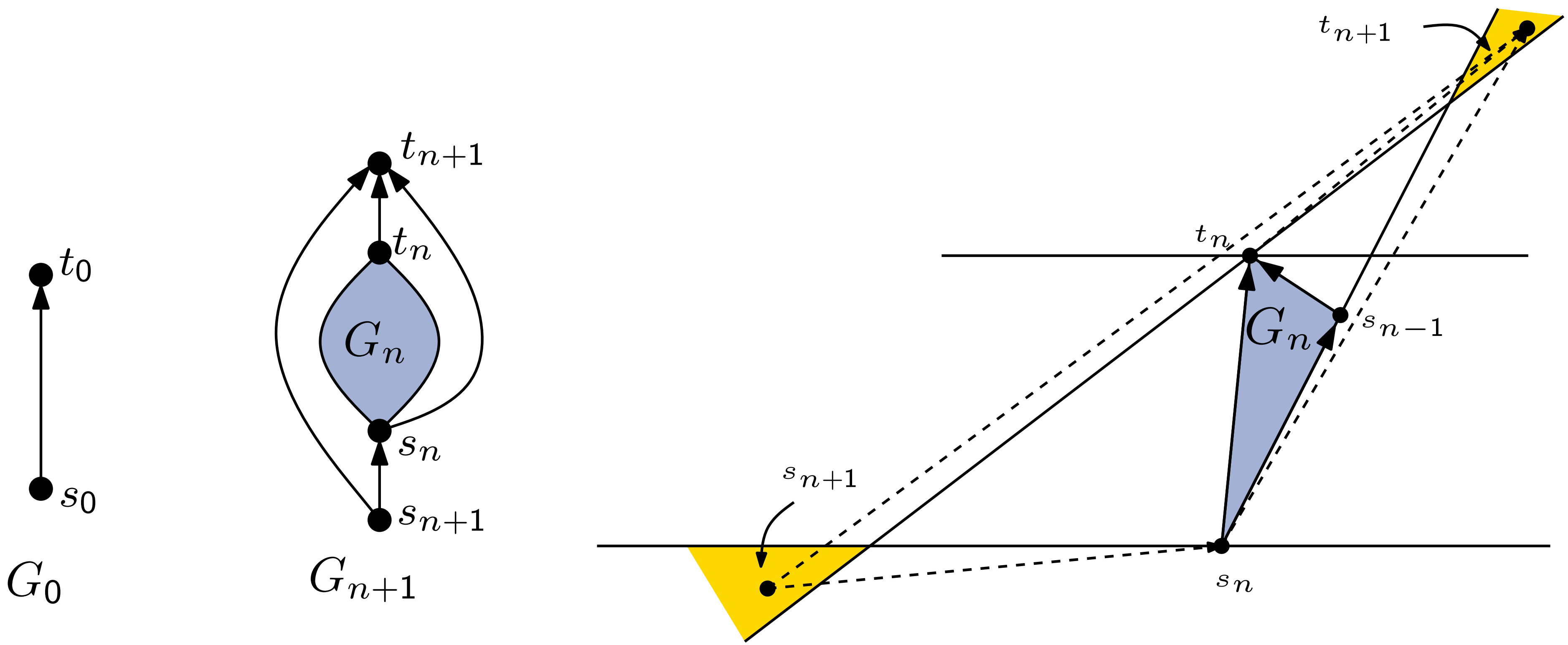


Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:

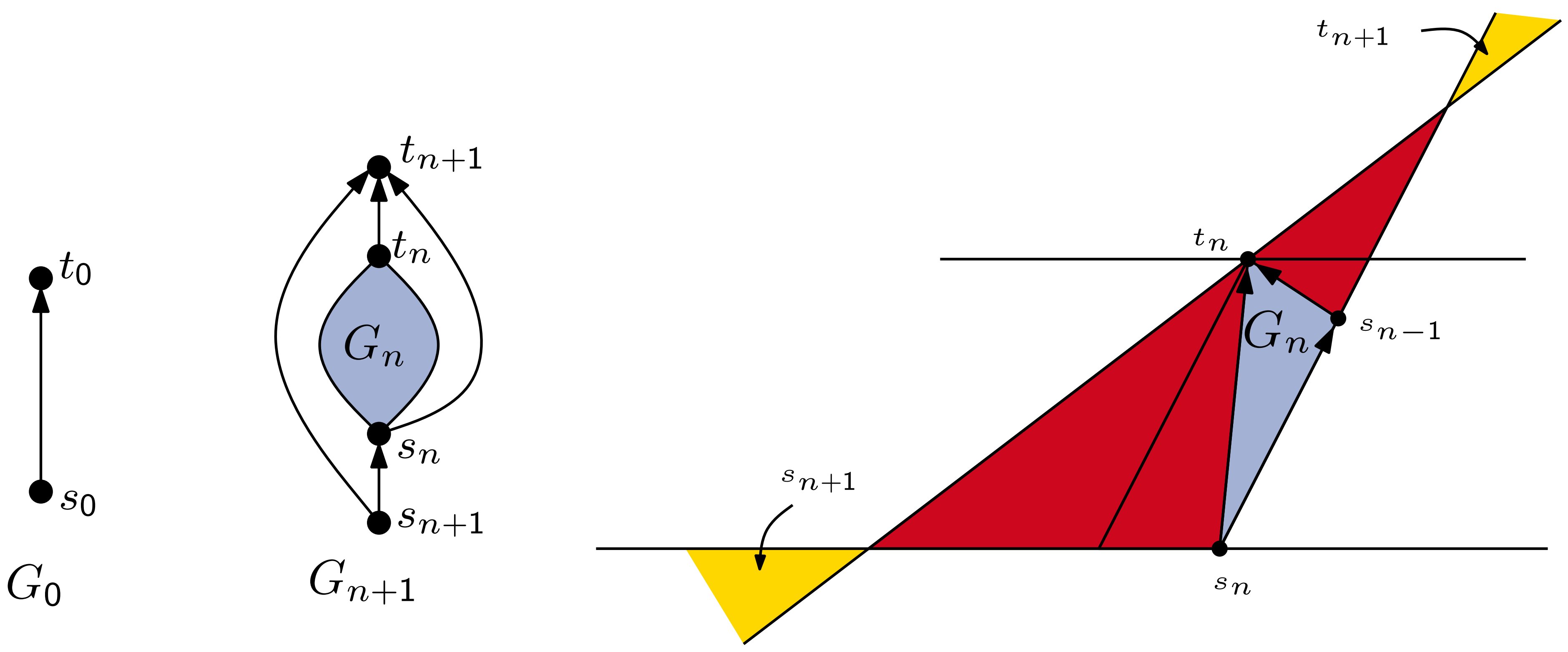


Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:

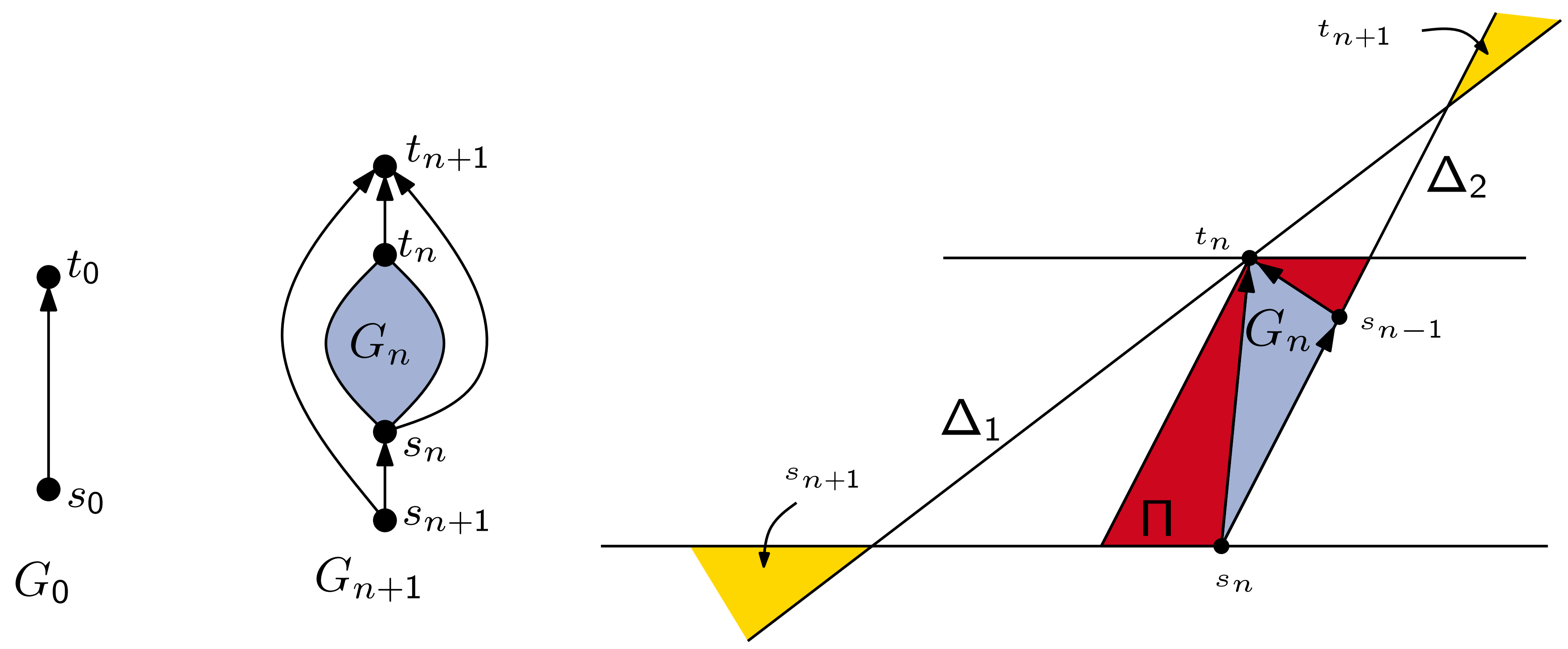


Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:



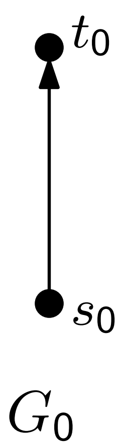
Lower Bound for the Area

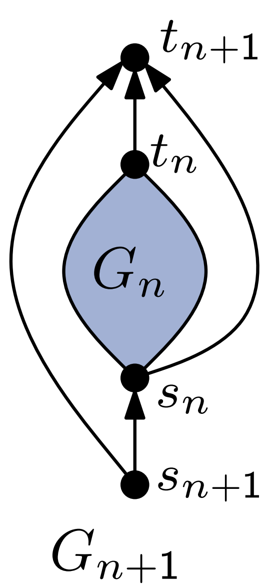
Theorem [Bertolazzi et al. 94]

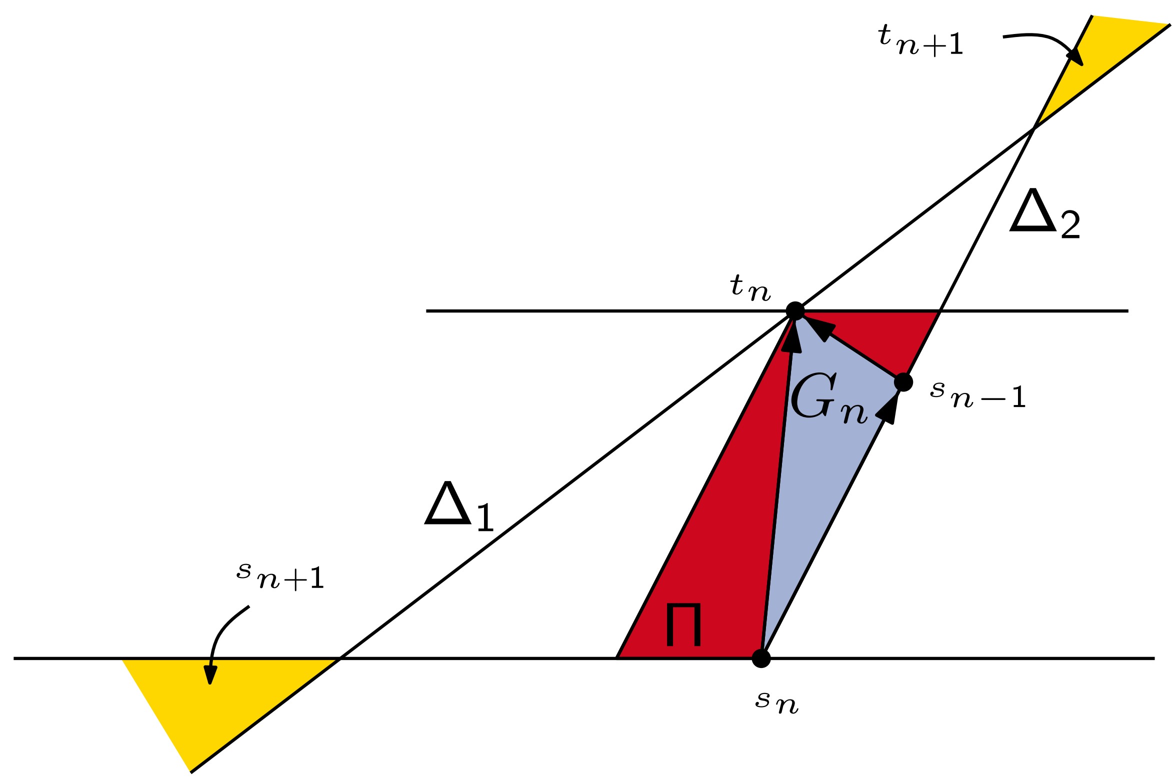
There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:

- We have that: $Area(\Pi) > 2 \cdot Area(G_n)$







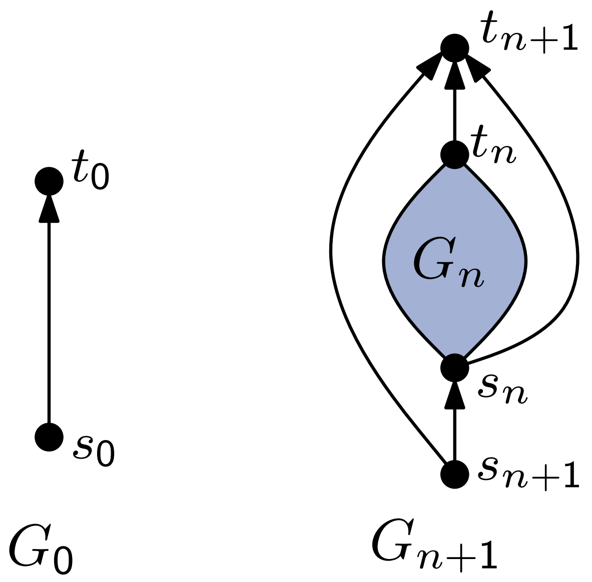
Lower Bound for the Area

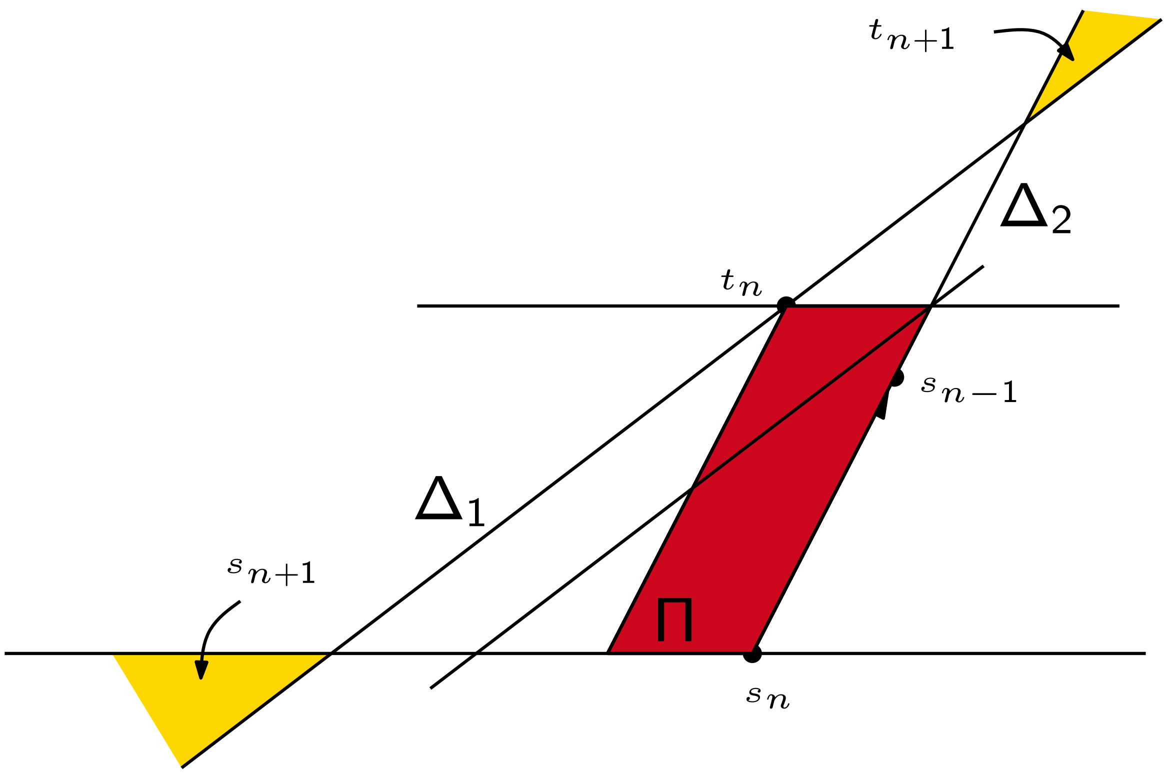
Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:

- We have that: $Area(\Pi) > 2 \cdot Area(G_n)$





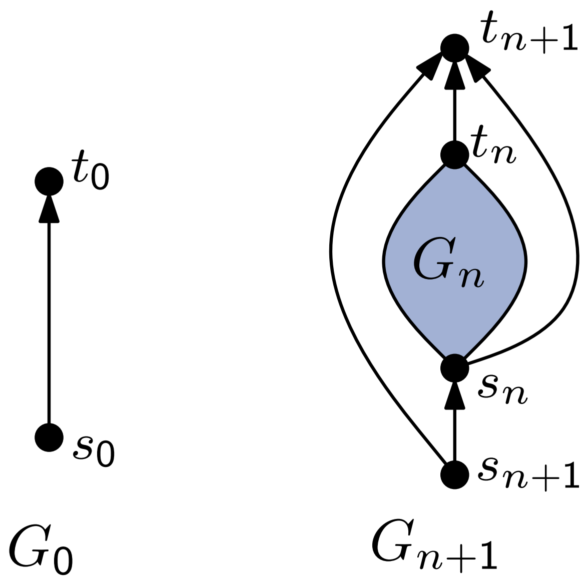
Lower Bound for the Area

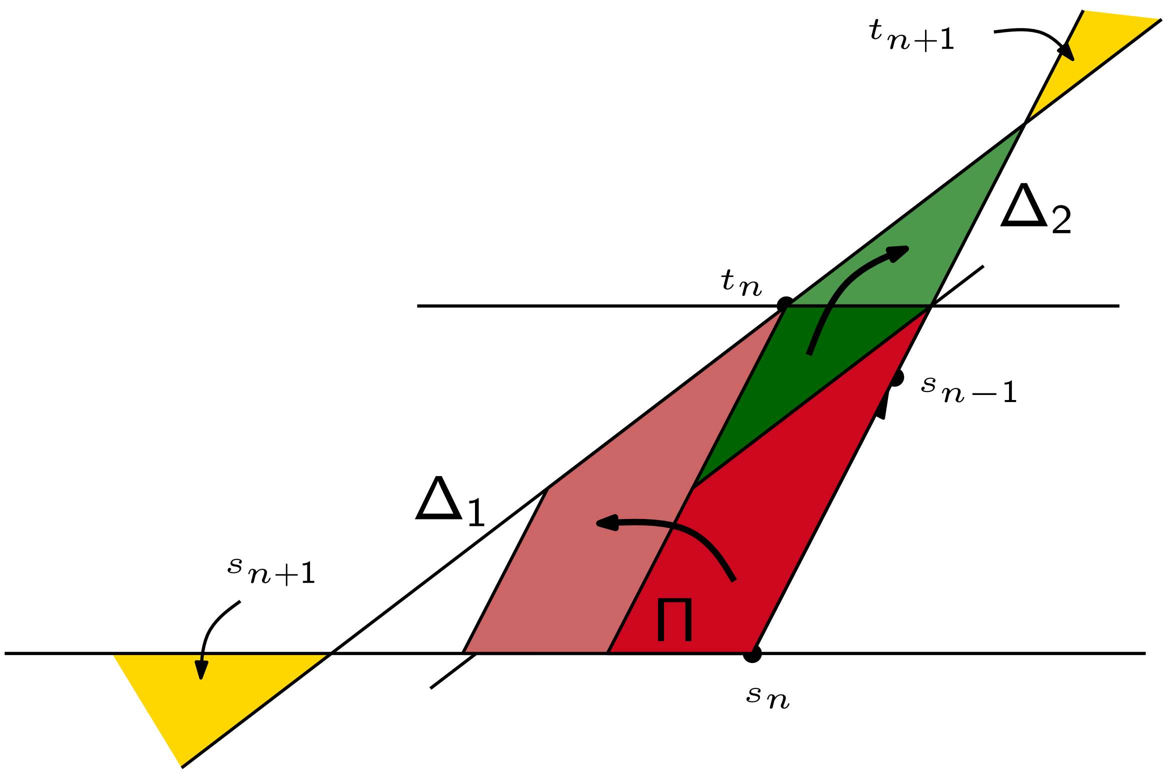
Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:

- We have that: $Area(\Pi) > 2 \cdot Area(G_n)$





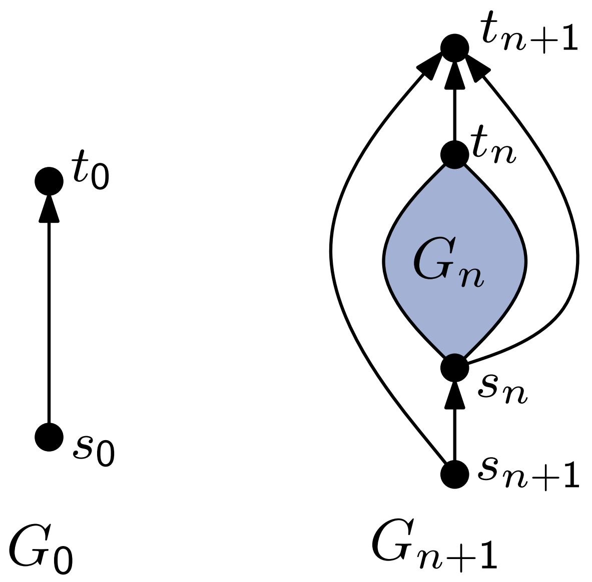
Lower Bound for the Area

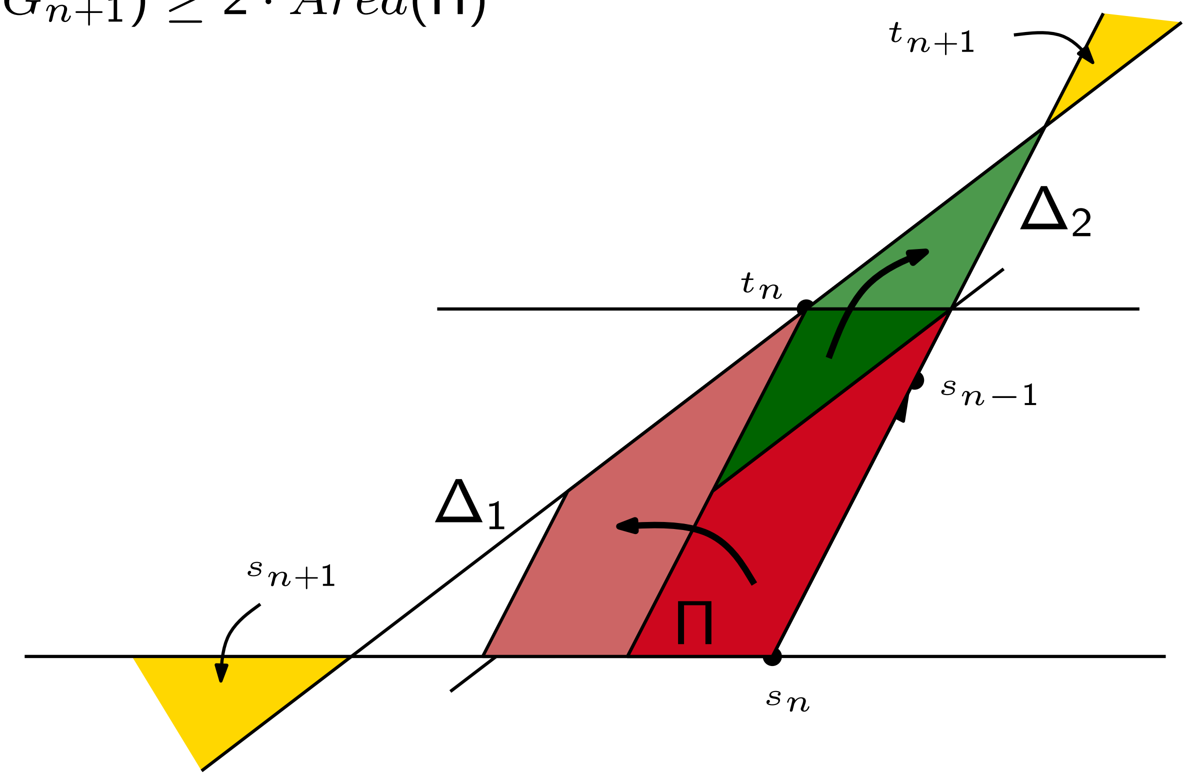
Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:

- We have that: $Area(\Pi) > 2 \cdot Area(G_n)$
- $Area(G_{n+1}) \geq 2 \cdot Area(\Pi)$





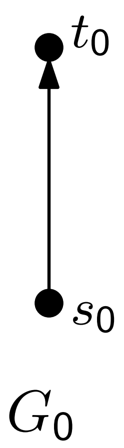
Lower Bound for the Area

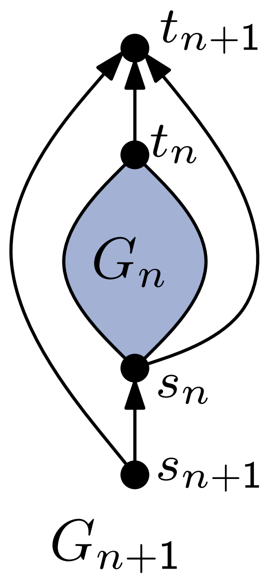
Theorem [Bertolazzi et al. 94]

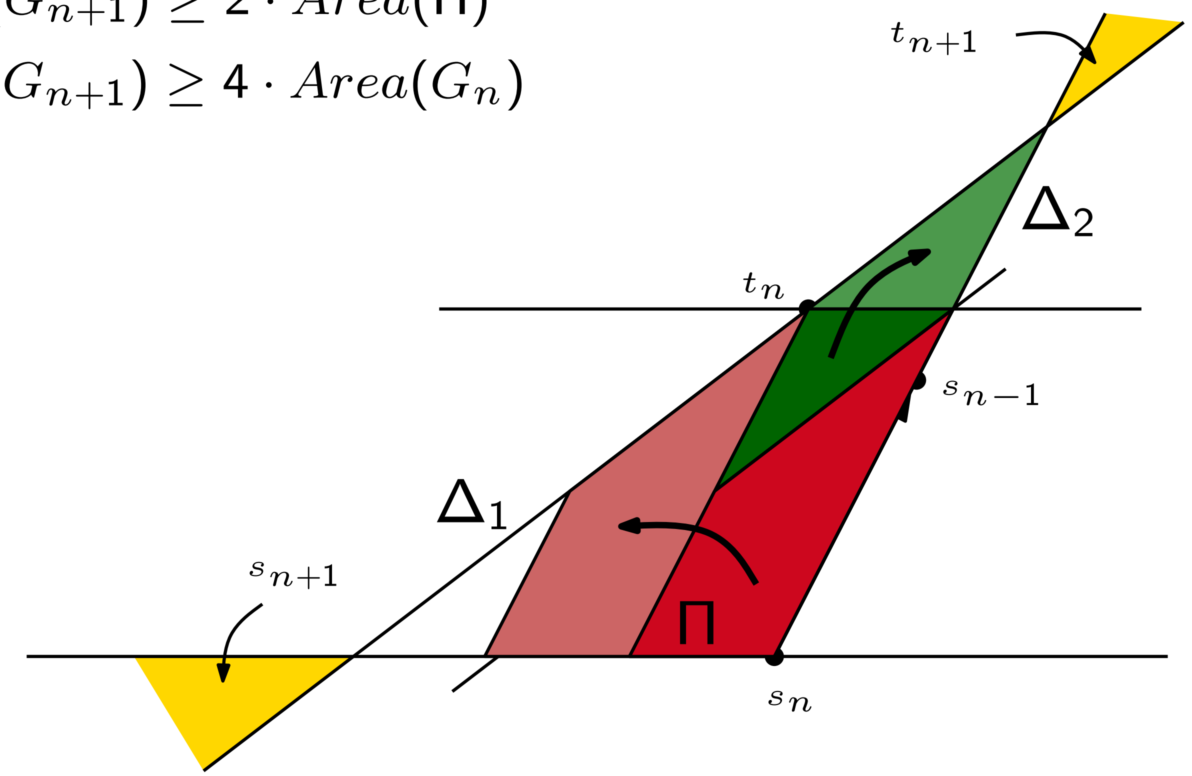
There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:

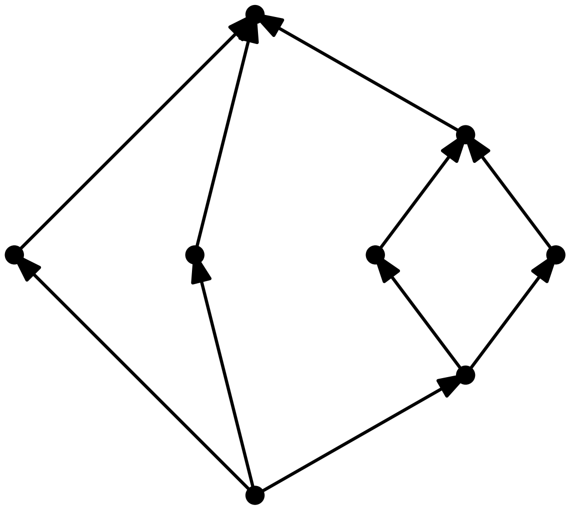
- We have that: $Area(\Pi) > 2 \cdot Area(G_n)$
- $Area(G_{n+1}) \geq 2 \cdot Area(\Pi)$
- $Area(G_{n+1}) \geq 4 \cdot Area(G_n)$

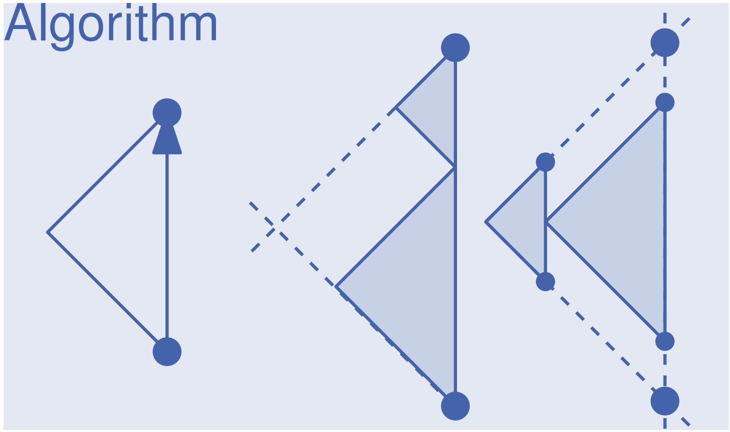


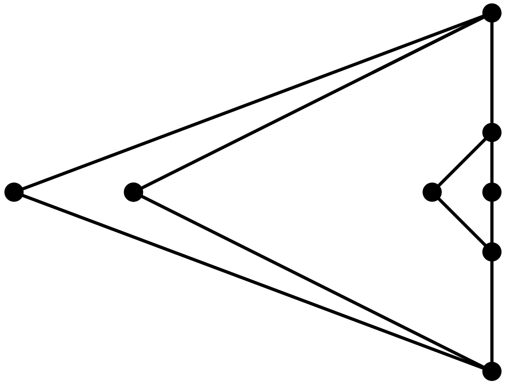




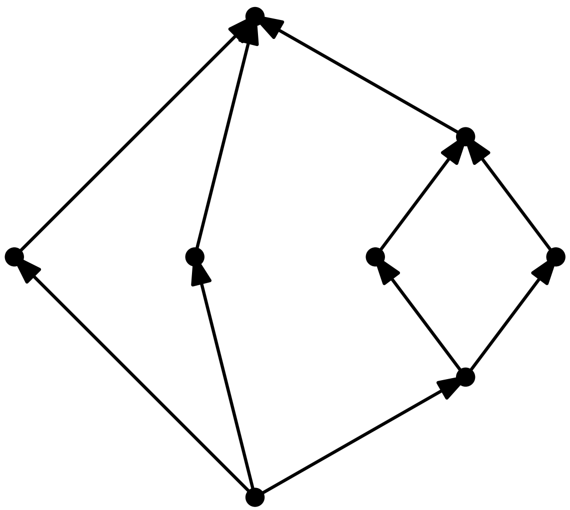
Property of the Algorithm

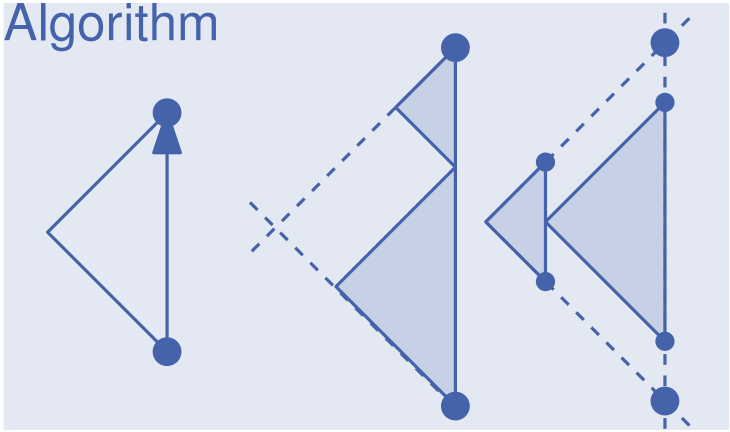


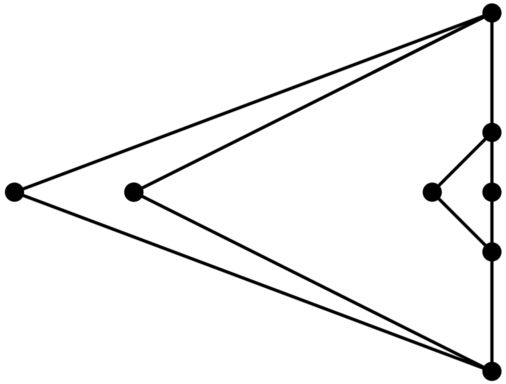


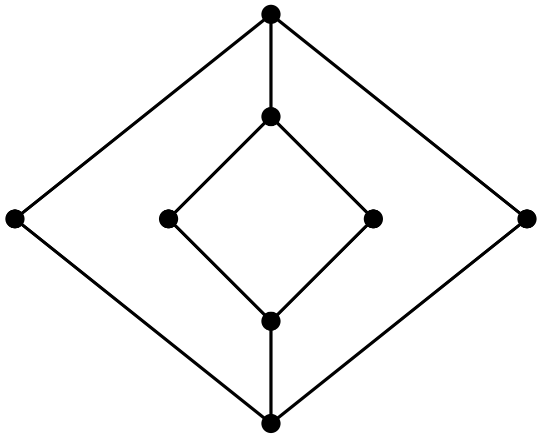


Property of the Algorithm



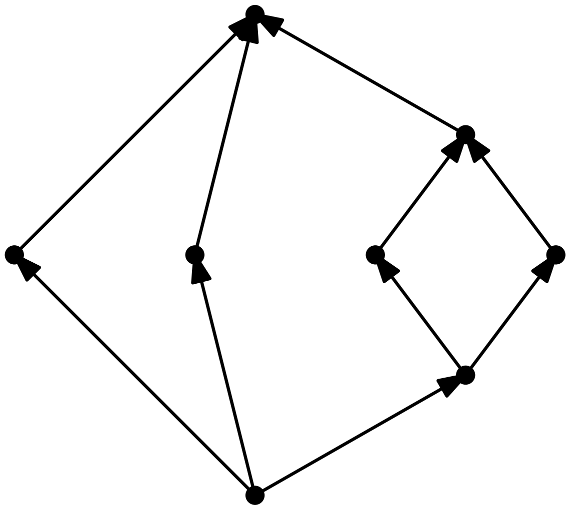


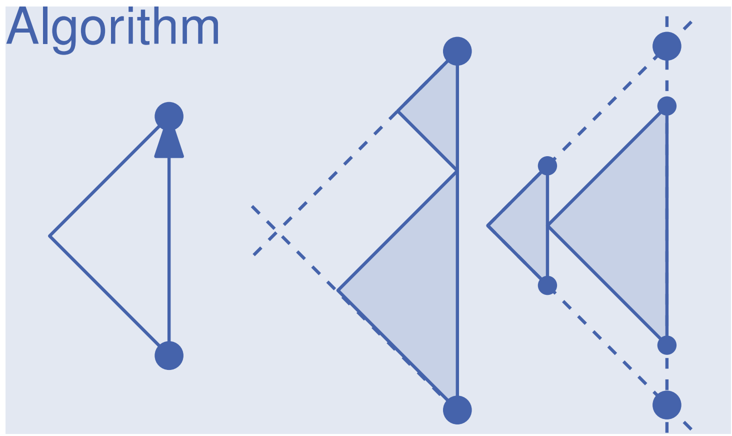


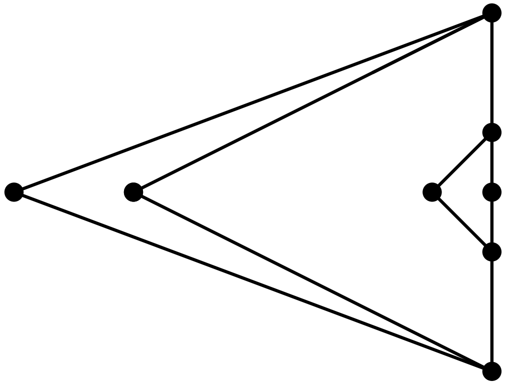


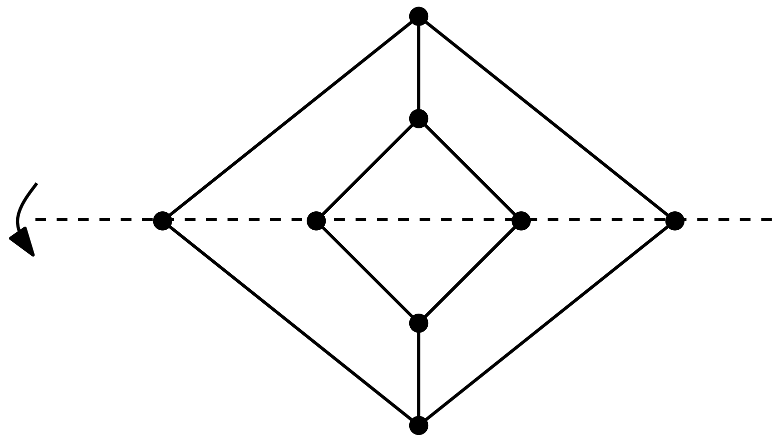
nicer???

Property of the Algorithm



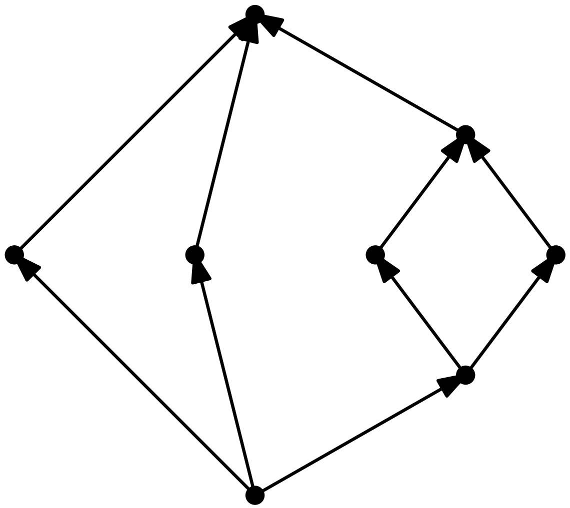


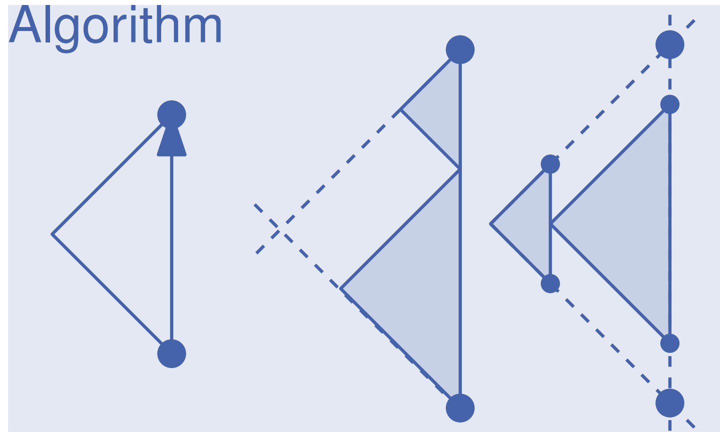


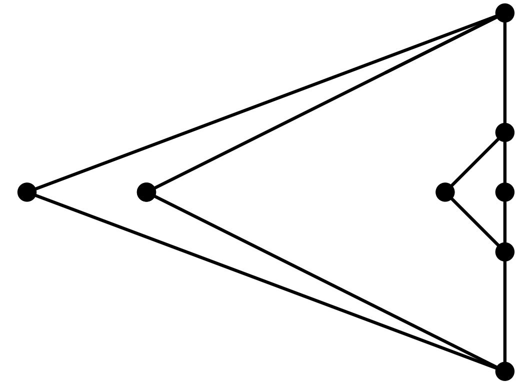


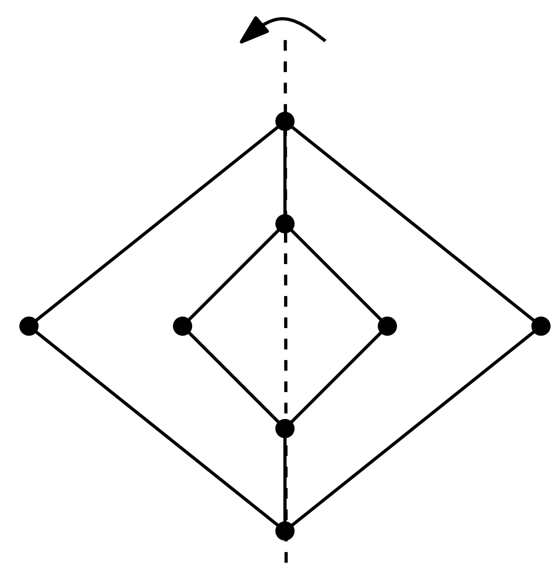
nicer???

Property of the Algorithm



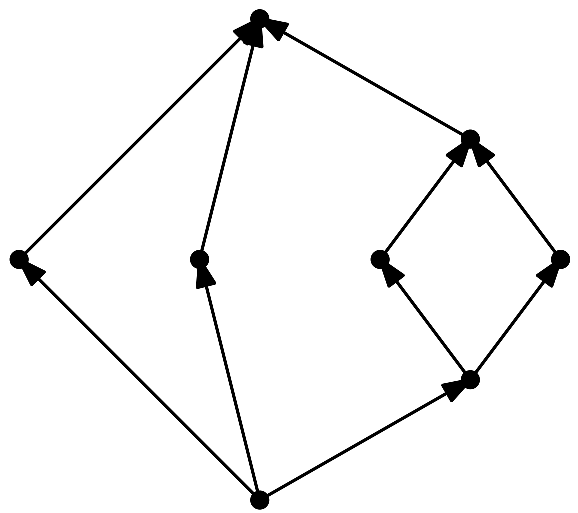


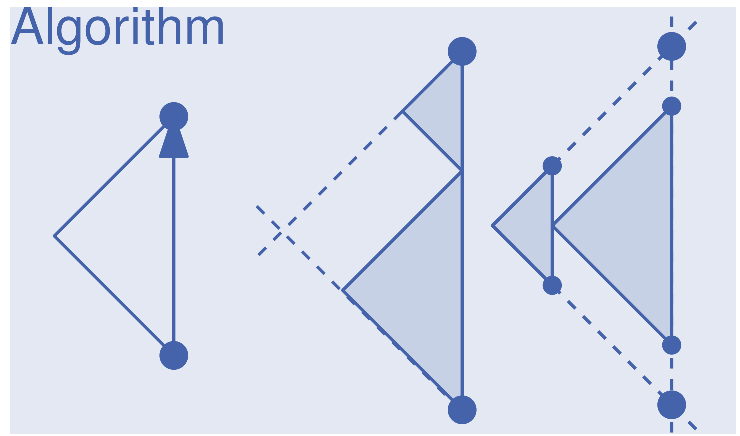


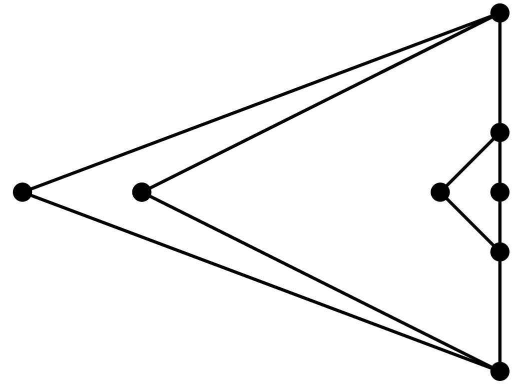


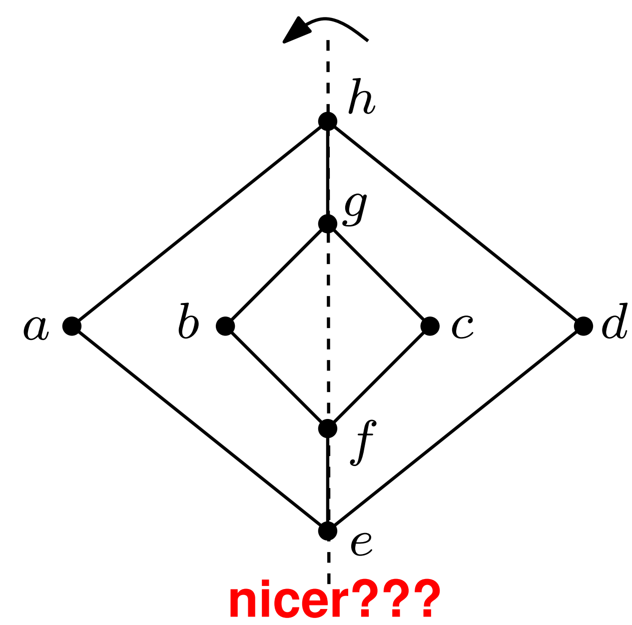
nicer???

Property of the Algorithm

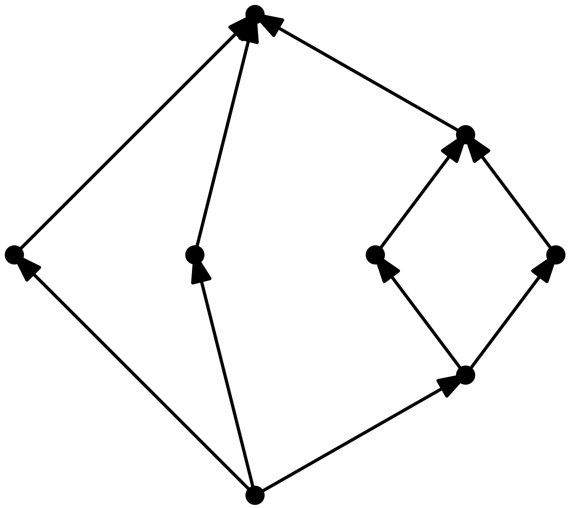


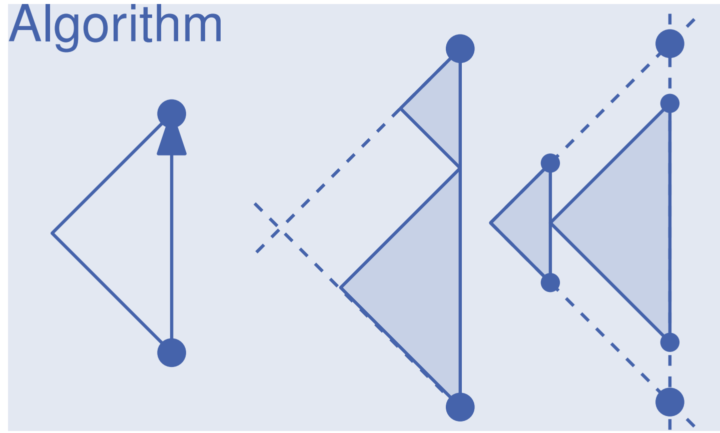


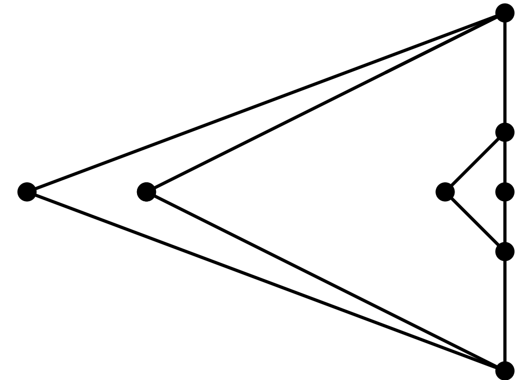




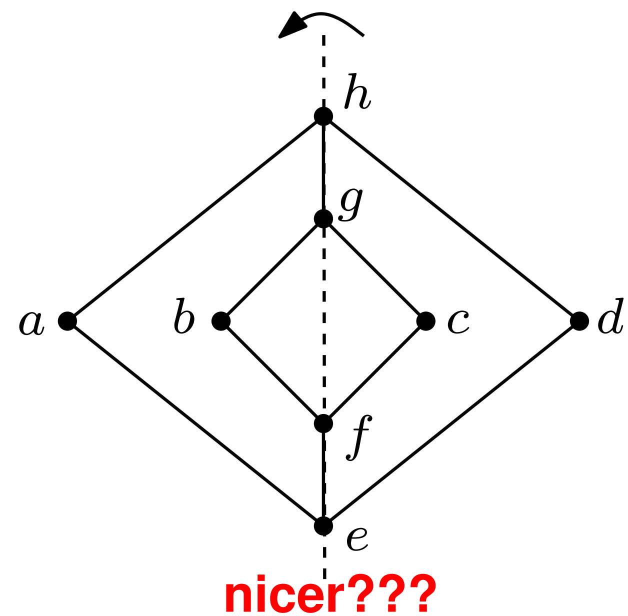
Property of the Algorithm



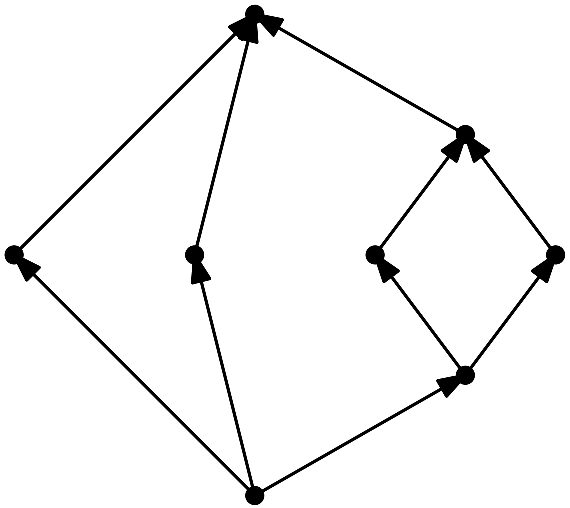


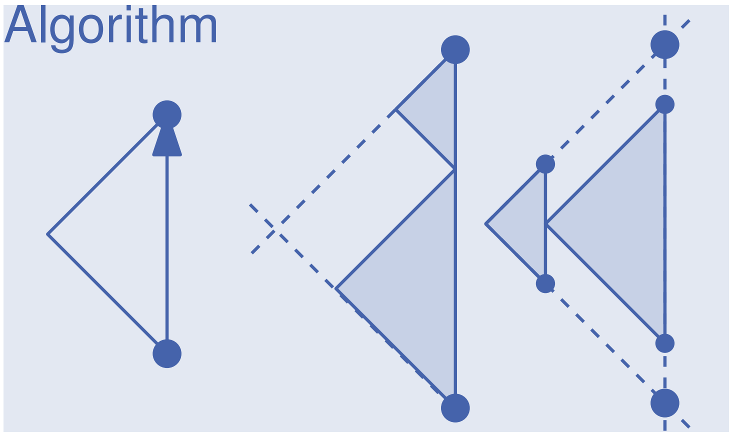


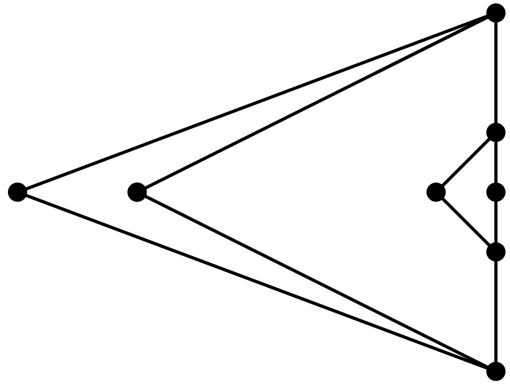
- Graph $G = (\{a, b, c, d, e, f, g, h\},$
  $\{(a, h),\ (a, e),\ (b, g),\ (b, f),\ (c, g),\ (c, f),\ (d, e),$
  $(d, h),\ (e, f),\ (h, g)\})$



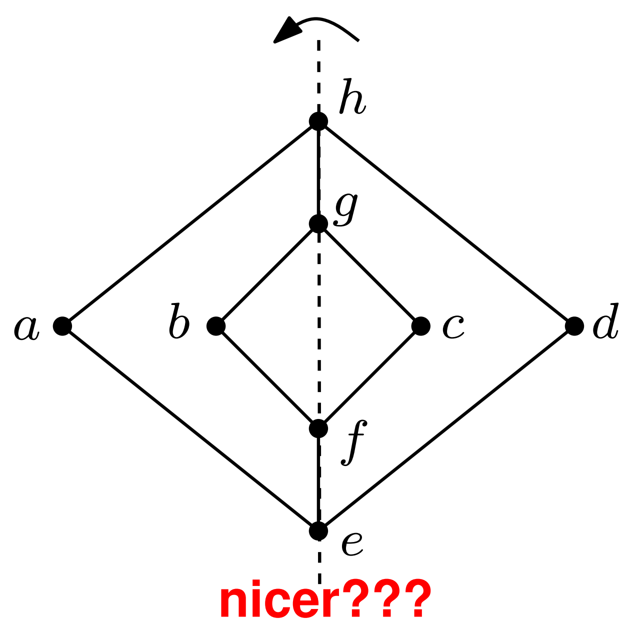
Property of the Algorithm



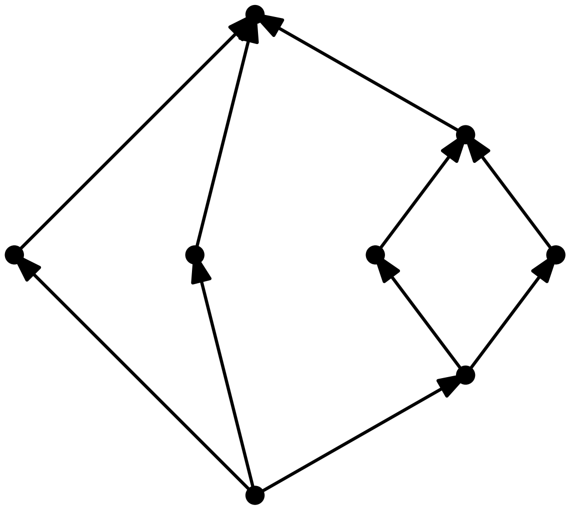


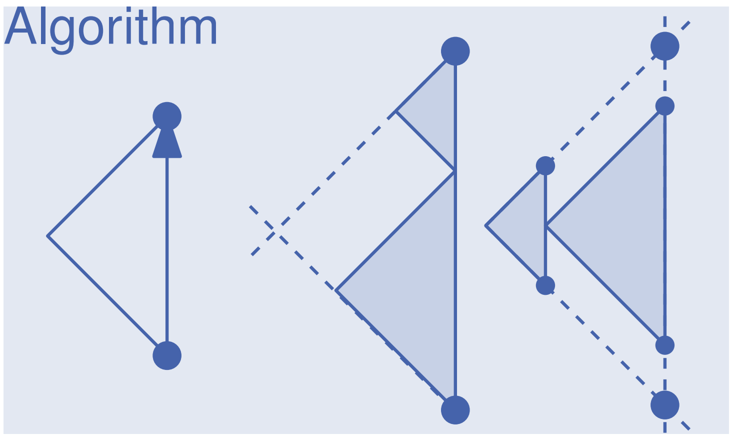


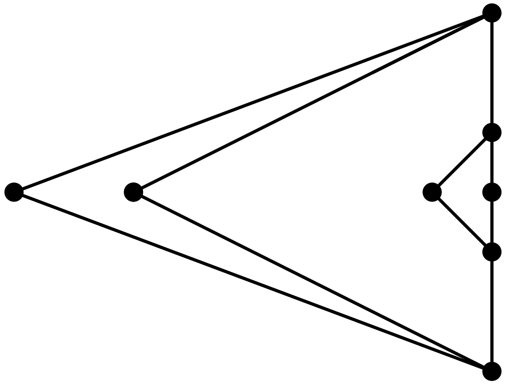
- Graph $G = (\{a, b, c, d, e, f, g, h\}, \{(a, h), (a, e), (b, g), (b, f), (c, g), (c, f), (d, e), (d, h), (e, f), (h, g)\})$
- Let G' be G where $b \to c \to b$, $a \to d \to a$.



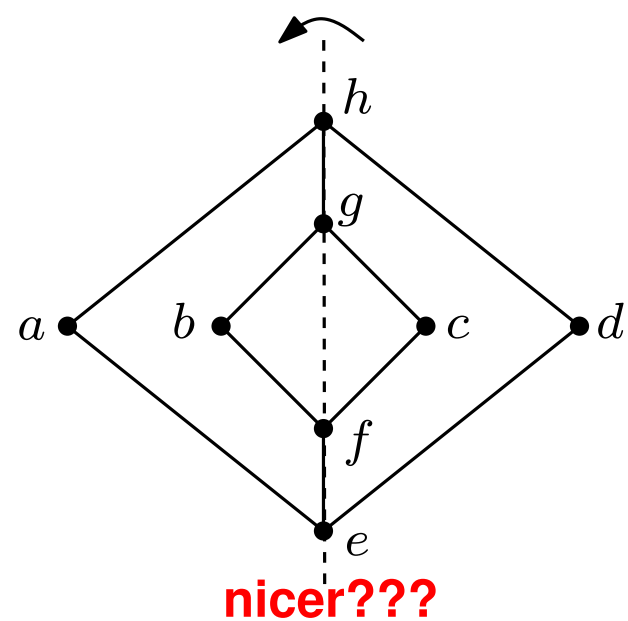
Property of the Algorithm







- Graph $G = (\{a, b, c, d, e, f, g, h\},$ $\{(a, h), (a, e), (b, g), (b, f), (c, g), (c, f), (d, e),$ $(d, h), (e, f), (h, g)\})$
- Let G' be G where $b \to c \to b$, $a \to d \to a$.
- G and G' are isomorphic.



Graph Automorphism

Definition: Automorphism of a digraph

An **automorphism** of a directed graph $G = (V, E)$ is a permutation of the vertex set which preserves adjacency of the vertices and either preserves or reverses all the directions of the edges:

- $(u, v) \in E \Leftrightarrow (\pi(u), \pi(v)) \in E$, or
- $(u, v) \in E \Leftrightarrow (\pi(v), \pi(u)) \in E$

Graph Automorphism

Karlsruhe Institute of Technology

Definition: Automorphism of a digraph

An **automorphism** of a directed graph $G = (V, E)$ is a permutation of the vertex set which preserves adjacency of the vertices and either preserves or reverses all the directions of the edges:

- $(u, v) \in E \Leftrightarrow (\pi(u), \pi(v)) \in E$, or
- $(u, v) \in E \Leftrightarrow (\pi(v), \pi(u)) \in E$

- The set of all automorphisms (direction preserving and reversing) forms the **automorphism group** of G.

Graph Automorphism

Definition: Automorphism of a digraph

An **automorphism** of a directed graph $G = (V, E)$ is a permutation of the vertex set which preserves adjacency of the vertices and either preserves or reverses all the directions of the edges:

- $(u, v) \in E \Leftrightarrow (\pi(u), \pi(v)) \in E$, or
- $(u, v) \in E \Leftrightarrow (\pi(v), \pi(u)) \in E$

- The set of all automorphisms (direction preserving and reversing) forms the **automorphism group** of G.
- Finding an automorphism group of a graph is *isomorphism complete*, that is equivalent to testing whether two graphs are isomorphic.

Graph Automorphism

Definition: Automorphism of a digraph

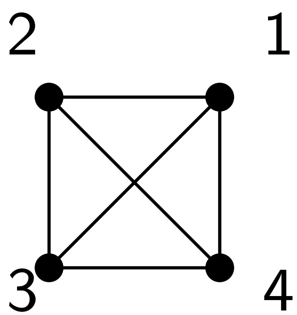
An **automorphism** of a directed graph $G = (V, E)$ is a permutation of the vertex set which preserves adjacency of the vertices and either preserves or reverses all the directions of the edges:

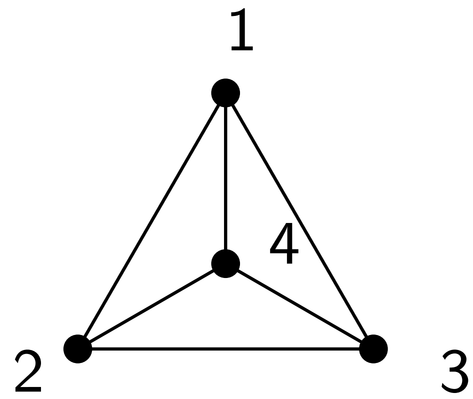
- $(u, v) \in E \Leftrightarrow (\pi(u), \pi(v)) \in E$, or
- $(u, v) \in E \Leftrightarrow (\pi(v), \pi(u)) \in E$

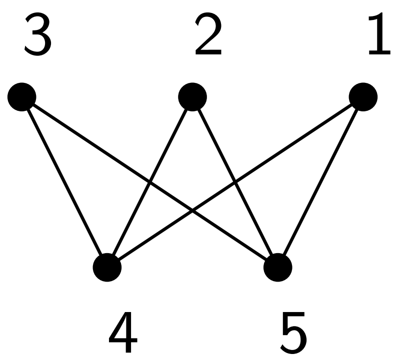
- The set of all automorphisms (direction preserving and reversing) forms the **automorphism group** of G.
- Finding an automorphism group of a graph is *isomorphism complete*, that is equivalent to testing whether two graphs are isomorphic.
- For planar graphs, graphs with bounded degree isomorphism problem has polynomial-time algorithms.

Geometric Automorphism

- Different types of automorphism:

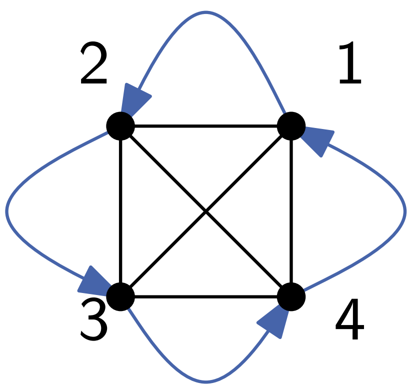




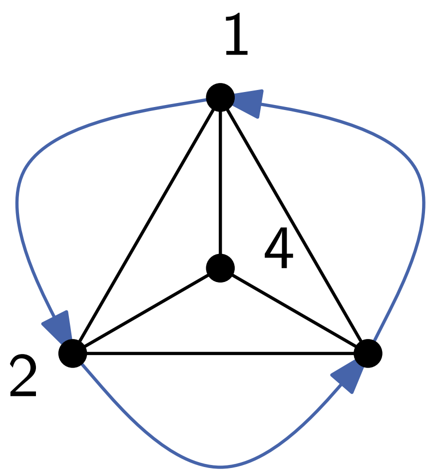


Geometric Automorphism

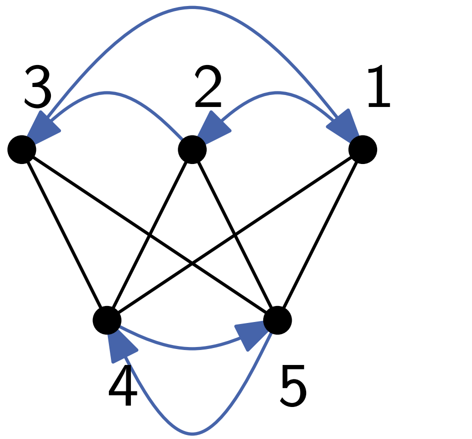
- Different types of automorphism:



Automorphism 1 $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ 4 $\rightarrow$ 1 is geometrically representable, while 1 $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ 1 is not.



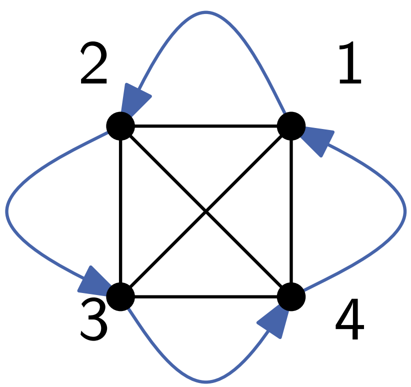
Automorphism 1 $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ 1 is geometrically representable, while 1 $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ 4 $\rightarrow$ 1 is not.



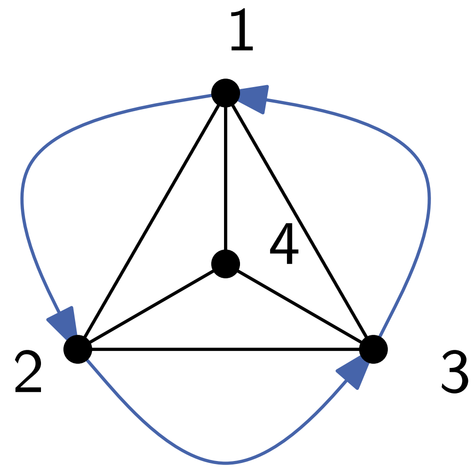
Automorphism 1 $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ 1, 4 $\rightarrow$ 5 $\rightarrow$ 4 is not geometrically representable.

Geometric Automorphism

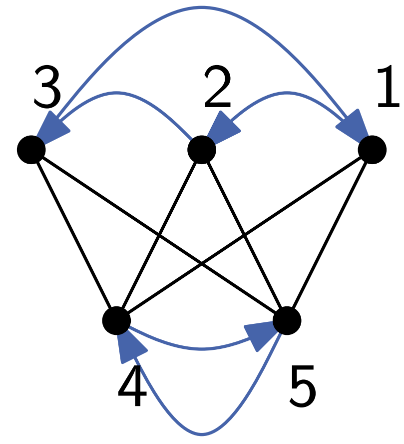
- Different types of automorphism:



Automorphism $1 \to 2 \to 3 \to 4 \to 1$ is geometrically representable, while $1 \to 2 \to 3 \to 1$ is not.



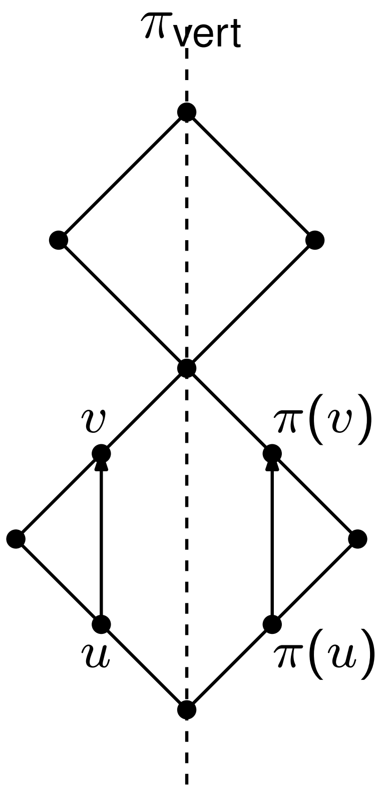
Automorphism $1 \to 2 \to 3 \to 1$ is geometrically representable, while $1 \to 2 \to 3 \to 4 \to 1$ is not.



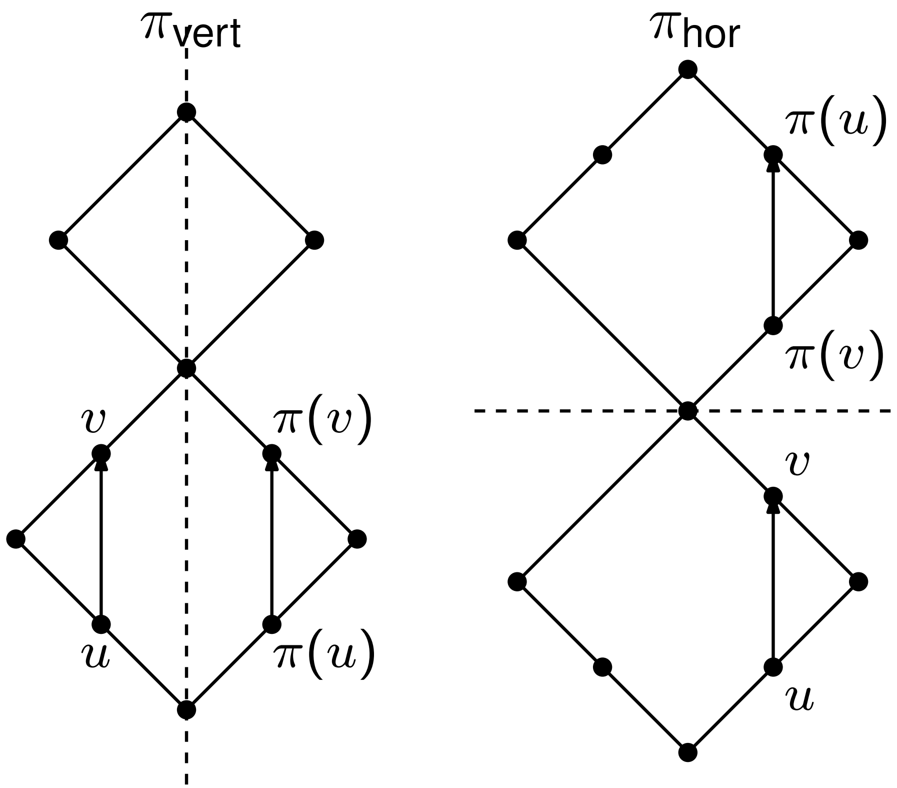
Automorphism $1 \to 2 \to 3 \to 1$, $4 \to 5 \to 4$ is not geometrically representable.

- An automorphism group P of a graph is geometric, if there exists a drawing of G that displays each element of P as a symmetry.
- For general graphs it is $\mathcal{NP}$-hard to find a geometric automorphism of a graph.
- For planar graphs, planar geometric automorphisms can be found in polynomial time. For outerplanar graphs and trees in linear time.

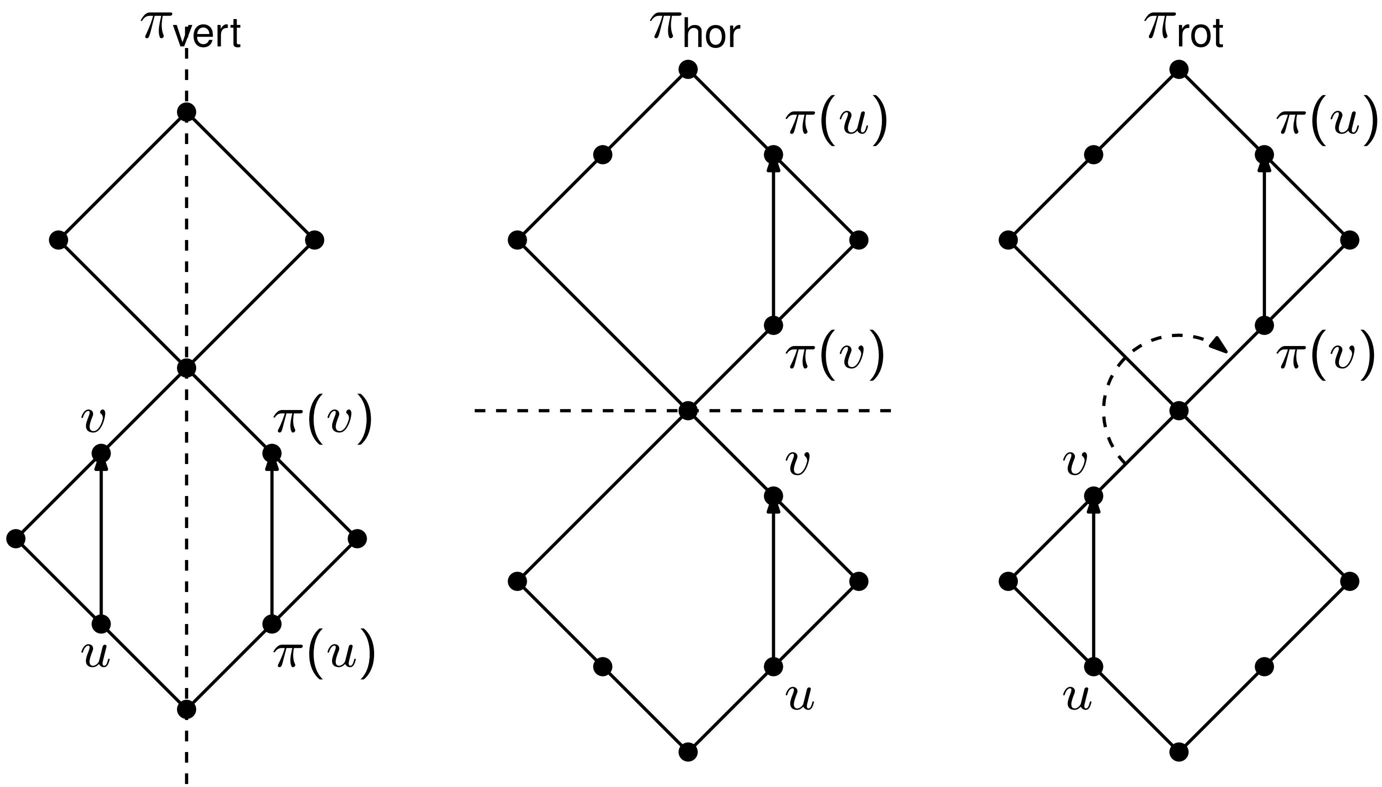
Symmetries in SP-Graphs



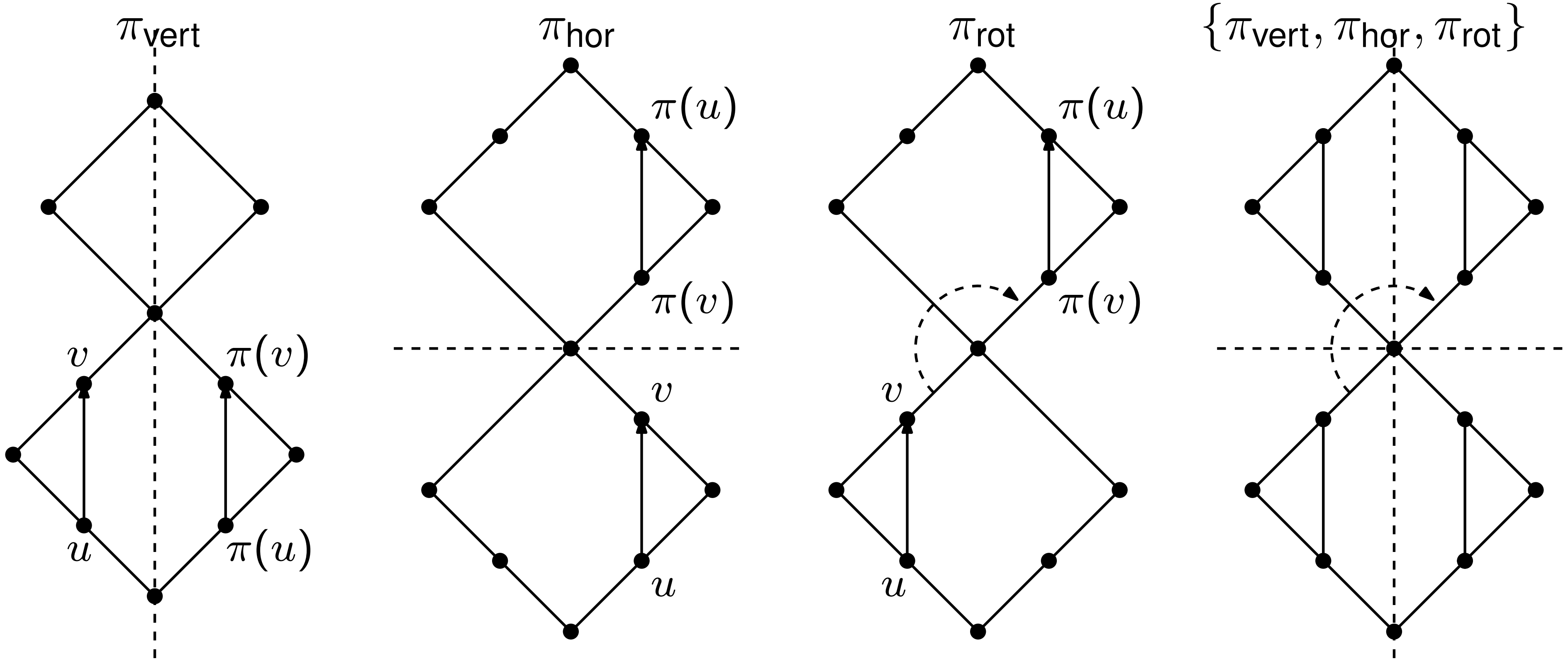
Symmetries in SP-Graphs



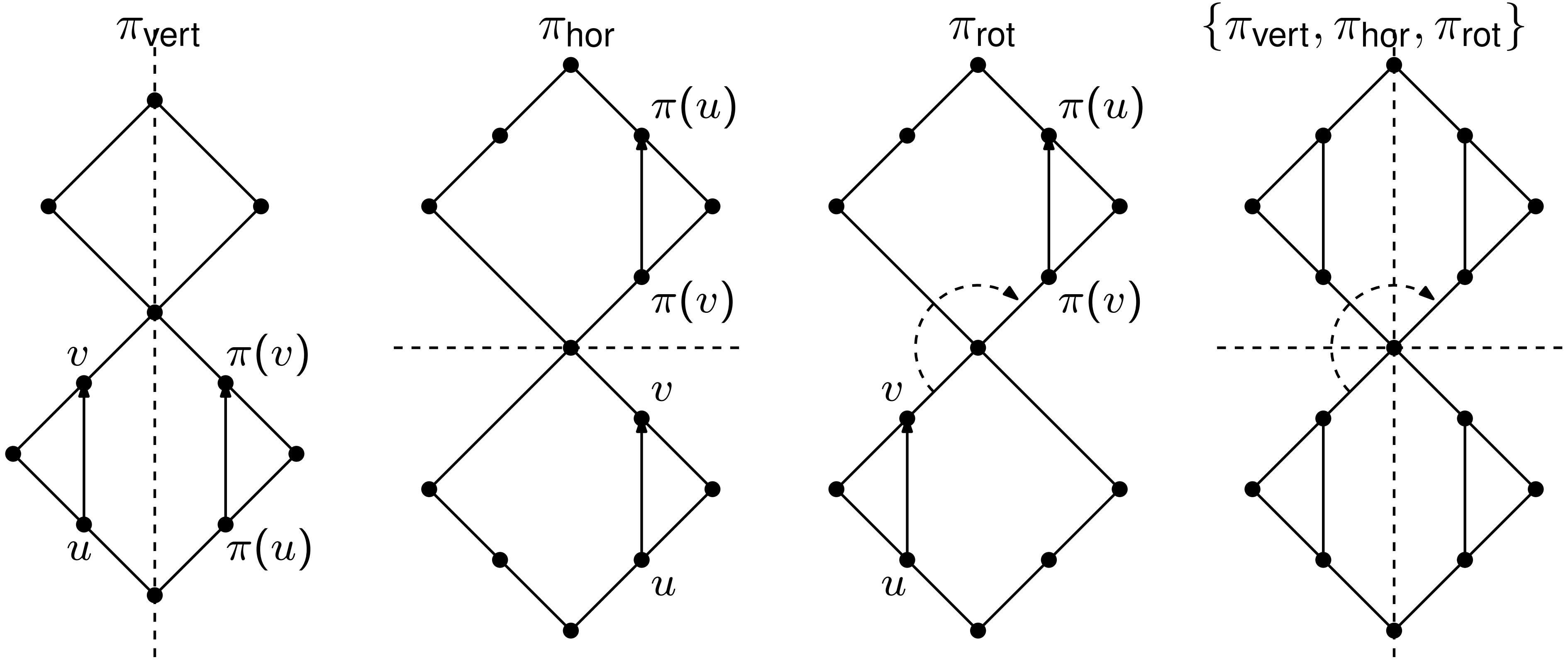
Symmetries in SP-Graphs



Symmetries in SP-Graphs

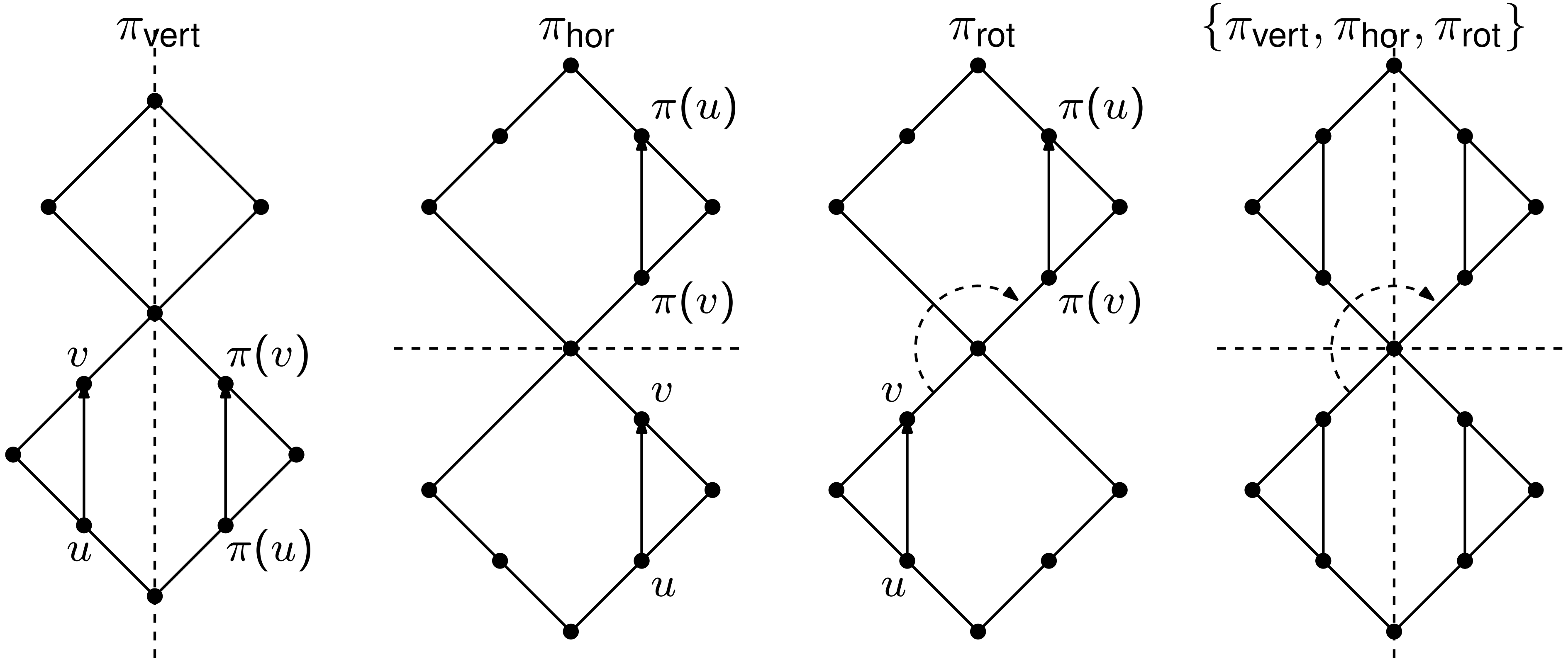


Symmetries in SP-Graphs



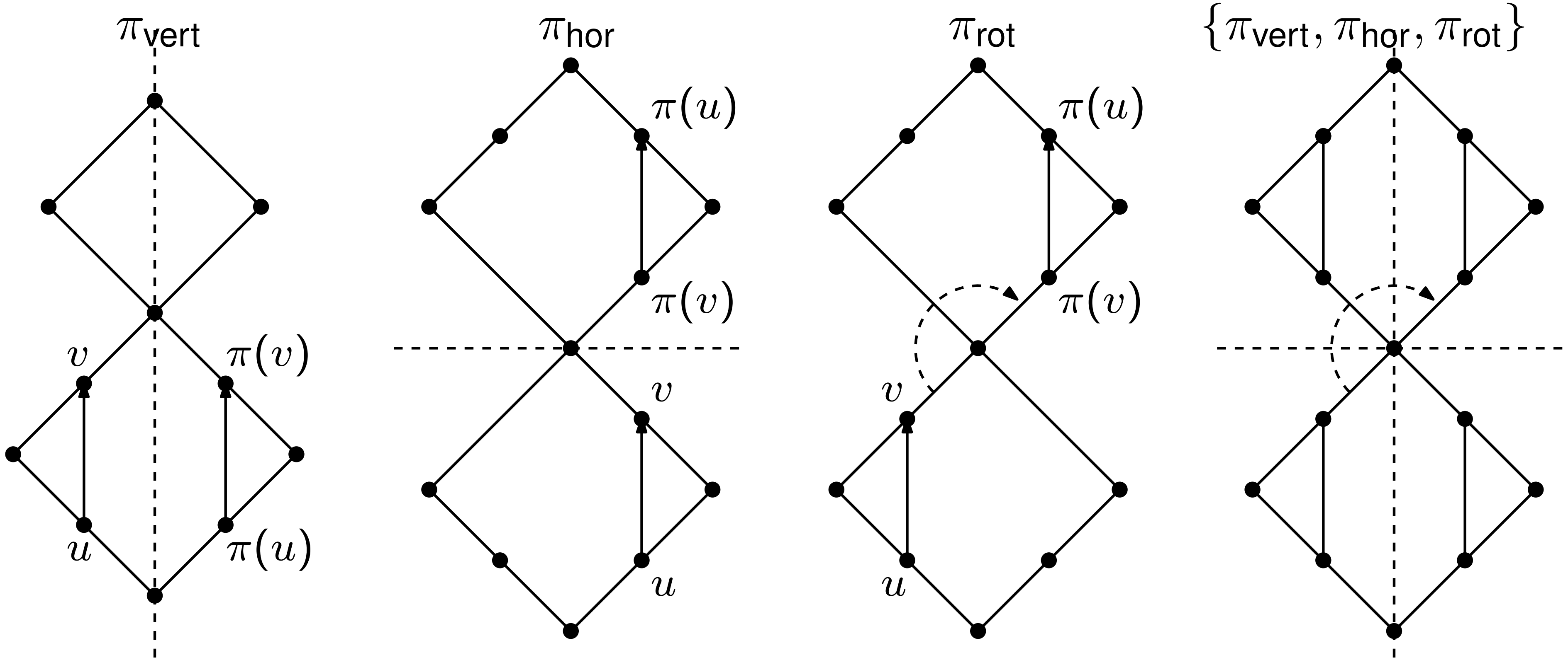
- A geometric automorphism group P of a graph G is upward planar, if there exists an upward planar drawing of G that displays each element of P as a symmetry.

Symmetries in SP-Graphs



- A geometric automorphism group P of a graph G is upward planar, if there exists an upward planar drawing of G that displays each element of P as a symmetry.
- How does a geometric automorphism group for a series-parallel graph look like?

Symmetries in SP-Graphs

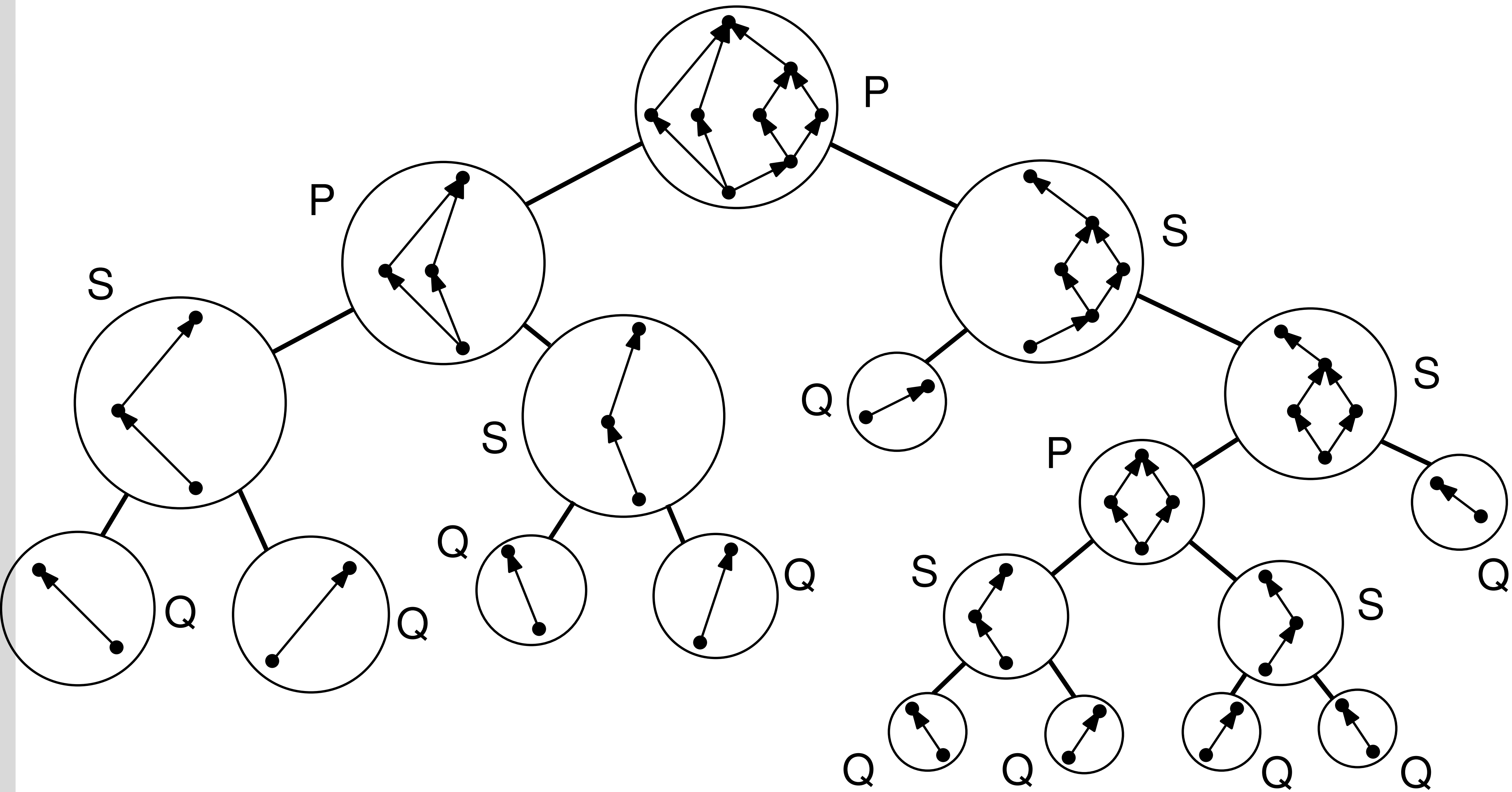


Theorem (Hong, Eades, Lee '00)

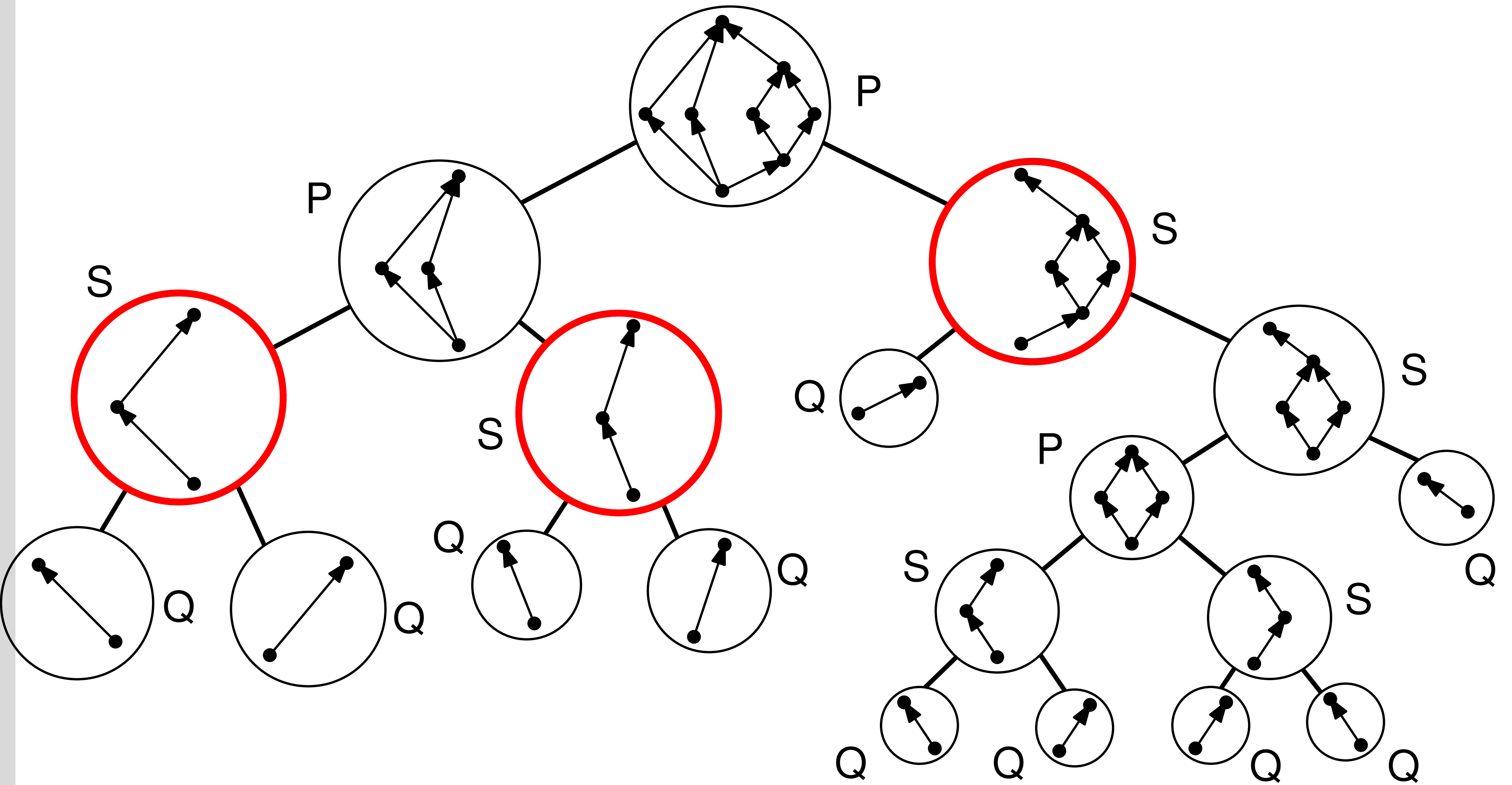
An upward planar automorphism group of a series-parallel digraph is either

- $\{\text{id}\}$
- $\{\text{id}, \pi\}$ with $\pi \in \{\pi_{\text{vert}}, \pi_{\text{hor}}, \pi_{\text{rot}}\}$
- $\{\text{id}, \pi_{\text{vert}}, \pi_{\text{hor}}, \pi_{\text{rot}}\}$.

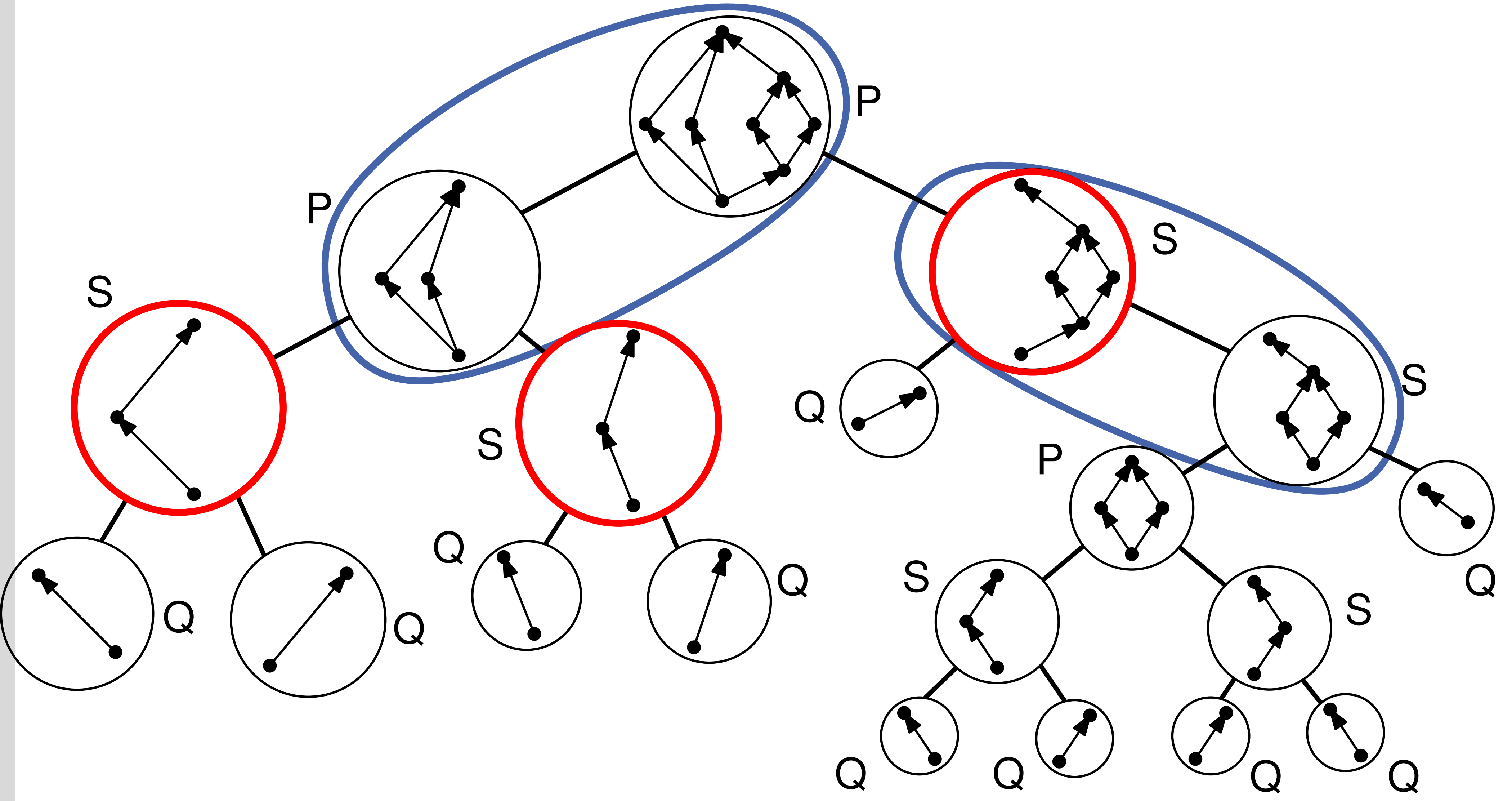
Vertical Automorphism



Vertical Automorphism

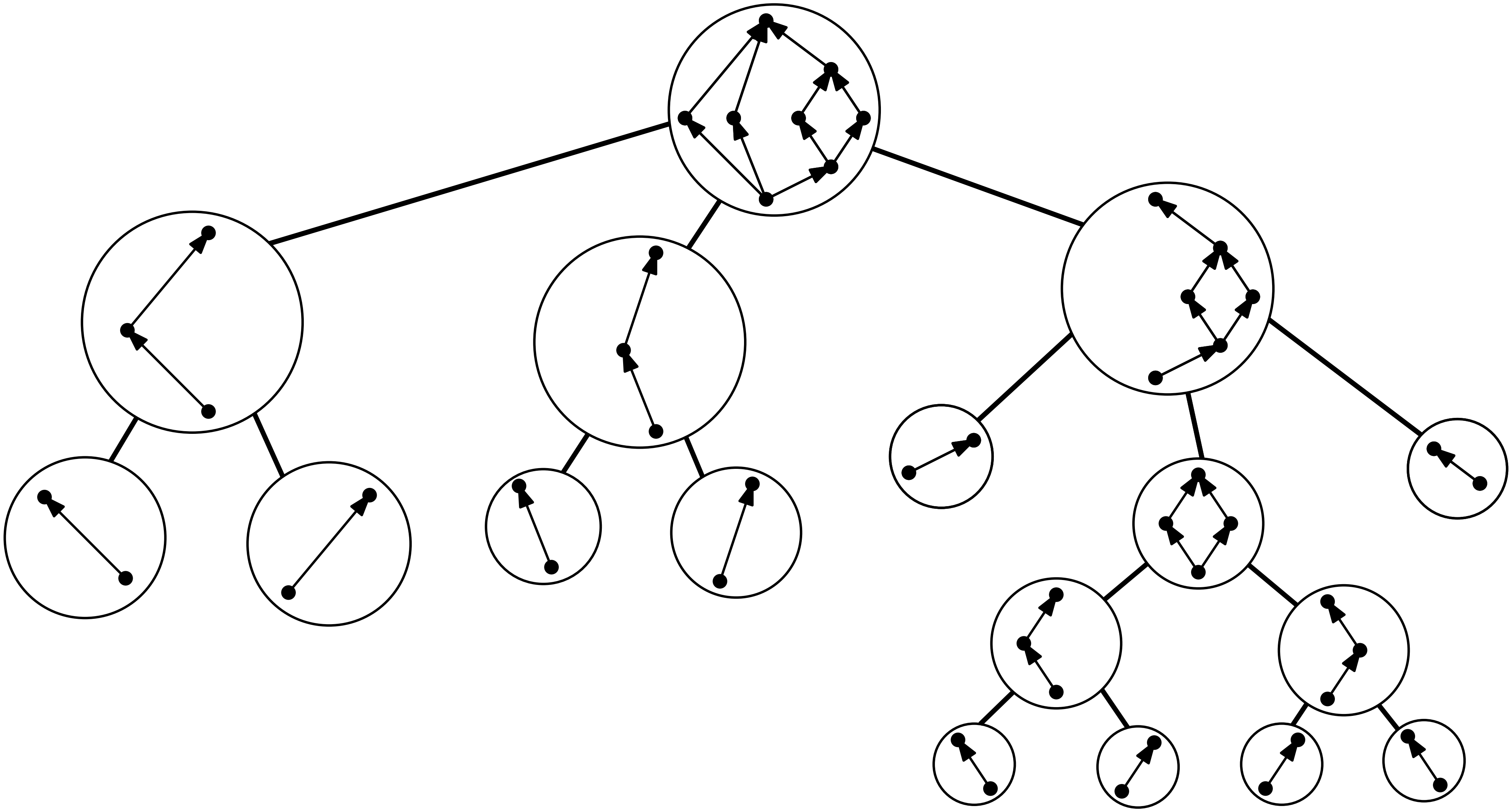


Vertical Automorphism



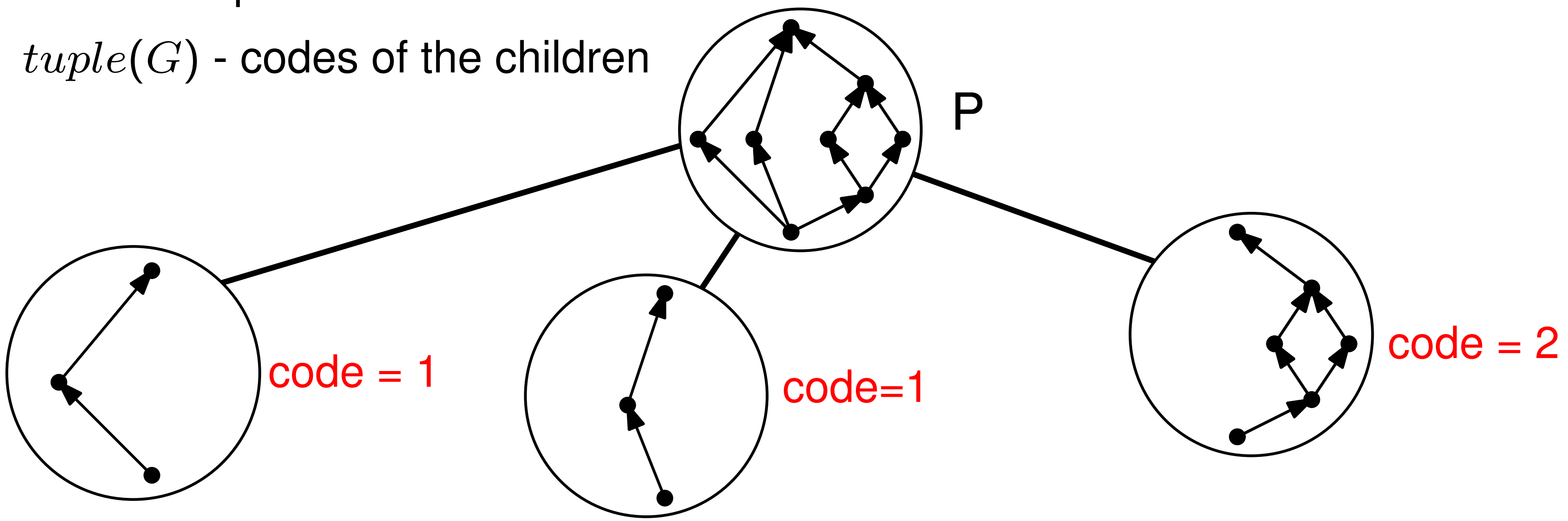
Institut für Theoretische Informatik
Lehrstuhl Algorithmik I

Vertical Automorphism



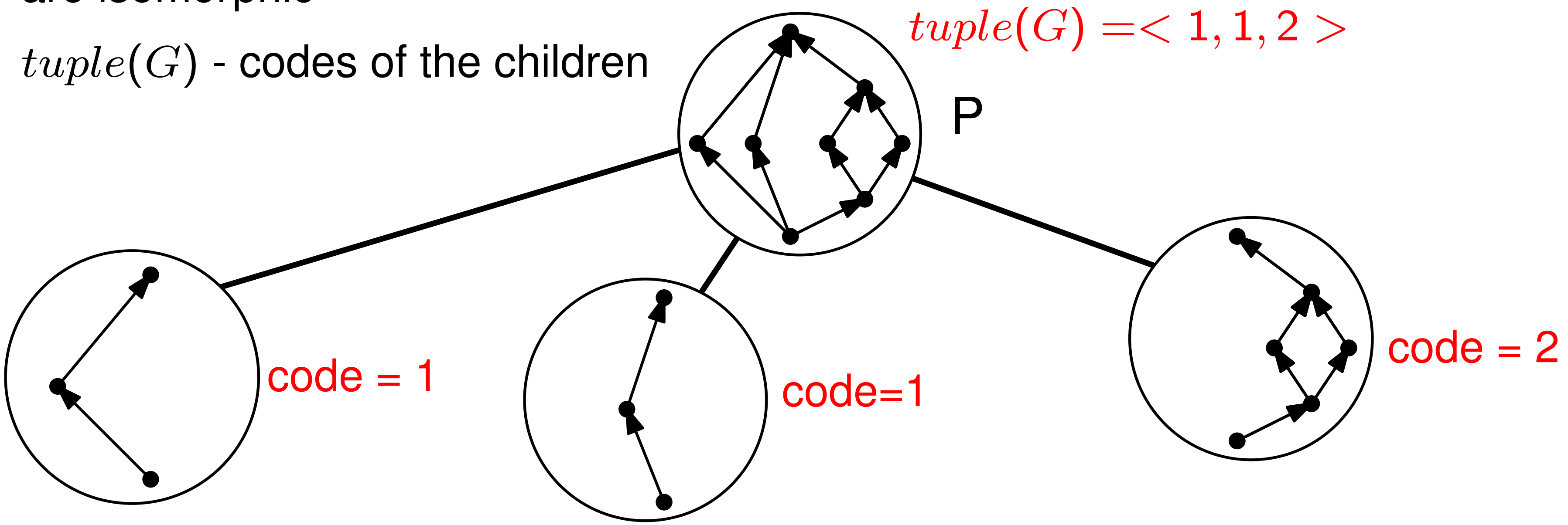
Vertical Automorphism

- $code(G)$ - two graphs at the same level have the same code iff they are isomorphic
- $tuple(G)$ - codes of the children



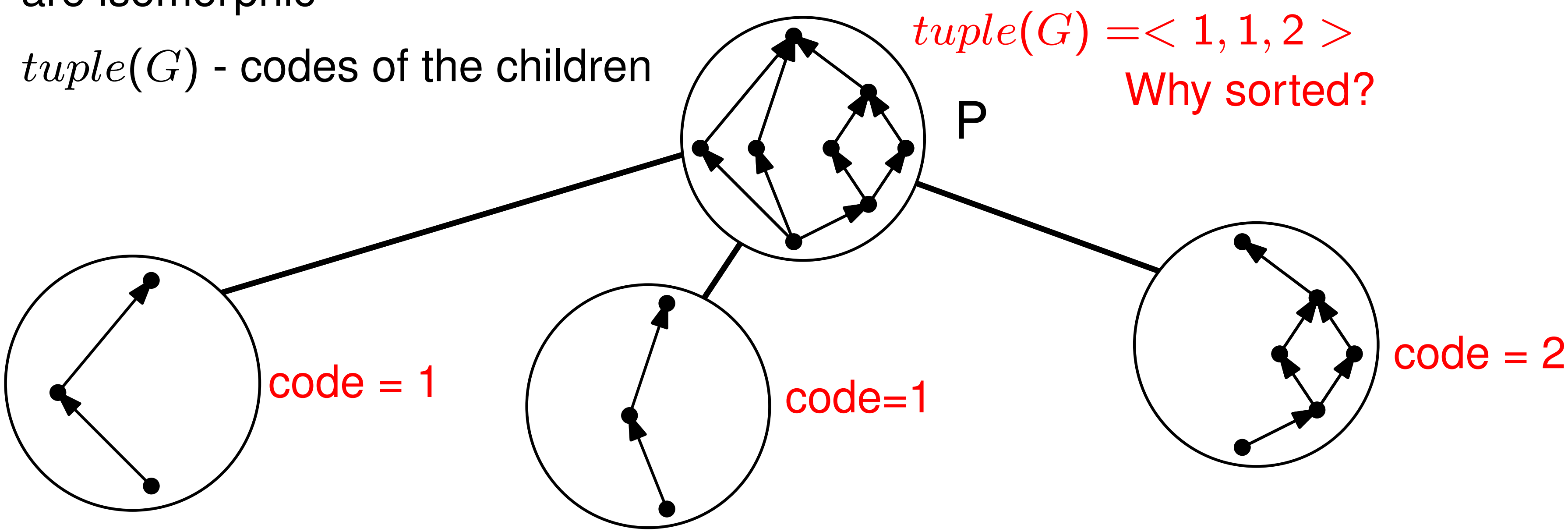
Vertical Automorphism

- $code(G)$ - two graphs at the same level have the same code iff they are isomorphic
- $tuple(G)$ - codes of the children



Vertical Automorphism

- $code(G)$ - two graphs at the same level have the same code iff they are isomorphic
- $tuple(G)$ - codes of the children

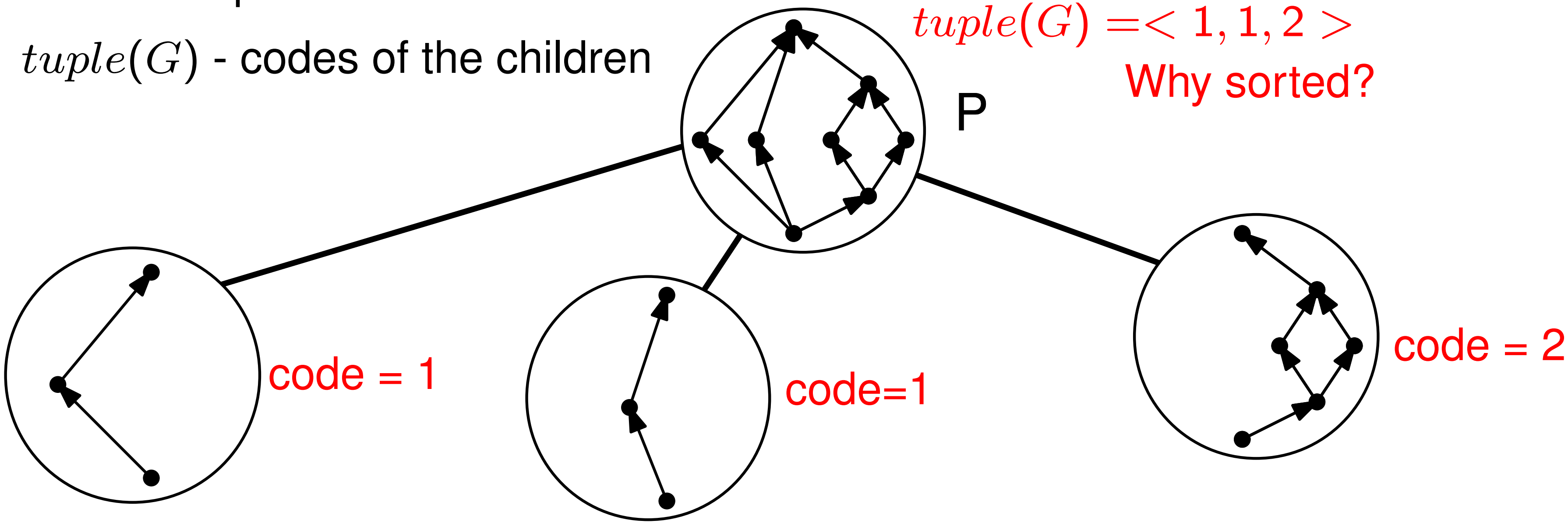


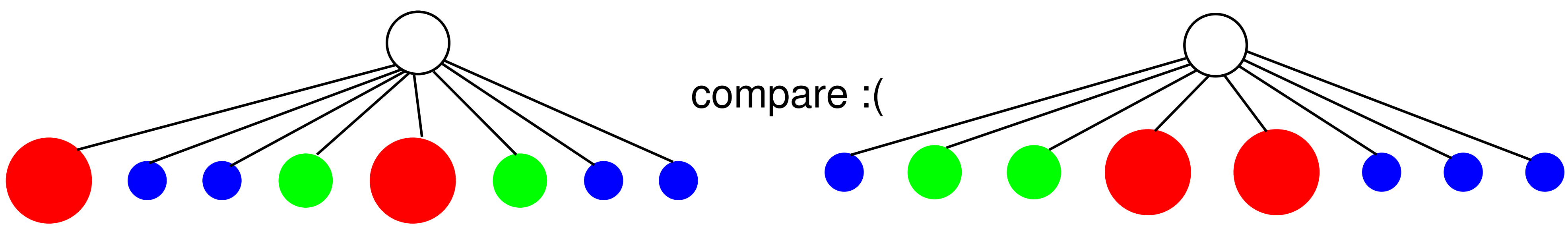
Vertical Automorphism

- $code(G)$ - two graphs at the same level have the same code iff they are isomorphic
- $tuple(G)$ - codes of the children

$tuple(G) =< 1, 1, 2 >$

Why sorted?





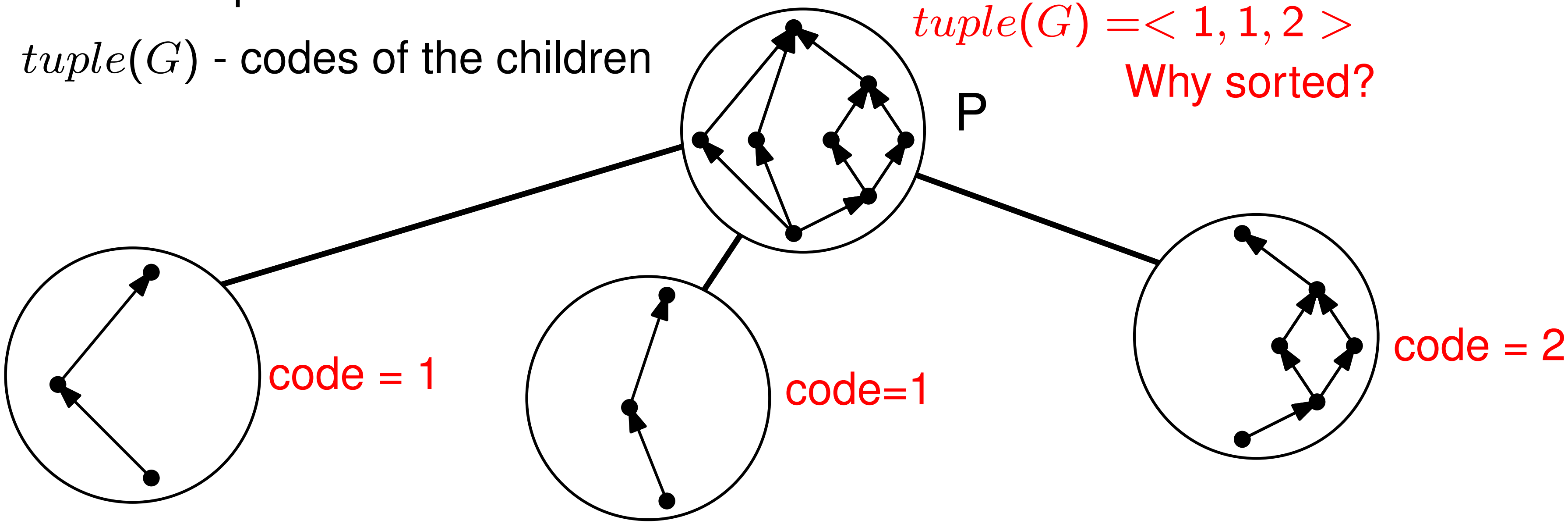
< 1 2 3 2 1 2 3 3 > < 3 2 2 1 1 3 3 3 >

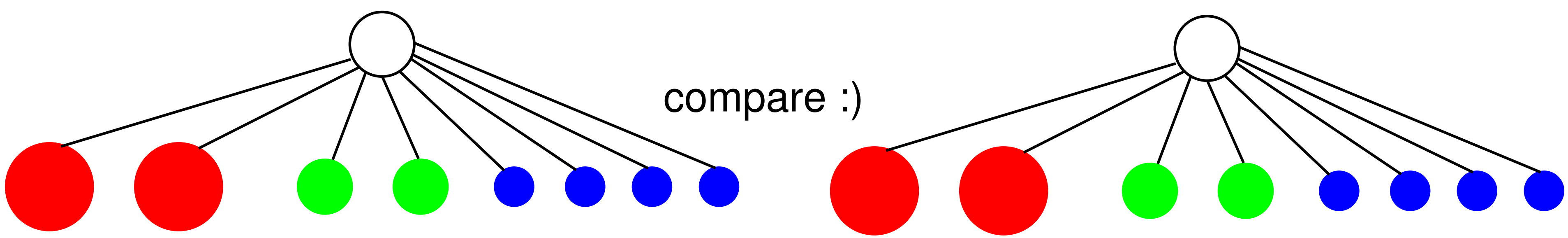
Vertical Automorphism

- $code(G)$ - two graphs at the same level have the same code iff they are isomorphic
- $tuple(G)$ - codes of the children

$tuple(G) =< 1, 1, 2 >$

Why sorted?





Vertical Automorphism

Algorithm constructing a Canonical Labeling

- Set $tuple(G_i) = \langle 0 \rangle$ for all Q-nodes G_i of G.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik I

Vertical Automorphism

Algorithm constructing a Canonical Labeling

- Set $tuple(G_i) = \langle 0 \rangle$ for all Q-nodes G_i of G.
- For each $t = \max \text{depth}(G), \ldots, 0$
  - For each S- or P-node G' at depth t with children $G_1, \ldots, G_k$ set $tuple(G') = \langle code(G_1), \ldots, code(G_k) \rangle$. If G' is a P-node, sort $tuple(G')$ in non-decreasing order.

Vertical Automorphism

Algorithm constructing a Canonical Labeling

- Set $tuple(G_i) = \langle 0 \rangle$ for all Q-nodes G_i of G.
- For each $t = \max \text{depth}(G), \ldots, 0$
  - For each S- or P-node G' at depth t with children $G_1, \ldots, G_k$ set $tuple(G') = \langle code(G_1), \ldots, code(G_k) \rangle$. If G' is a P-node, sort $tuple(G')$ in non-decreasing order.
  - Sort all the nodes at depth t lexicografically according to tuples.

Vertical Automorphism

Karlsruhe Institute of Technology

Algorithm constructing a Canonical Labeling

- Set $tuple(G_i) = \langle 0 \rangle$ for all Q-nodes G_i of G.
- For each $t = \max \text{depth}(G), \ldots, 0$
  - For each S- or P-node G' at depth t with children $G_1, \ldots, G_k$ set $tuple(G') = \langle code(G_1), \ldots, code(G_k) \rangle$. If G' is a P-node, sort $tuple(G')$ in non-decreasing order.
  - Sort all the nodes at depth t lexicografically according to tuples.
  - For each component G' at depth t, compute $code(G')$ as follows. Assign the integer 1 to those components represented by the first distinct tuple, assign 2 to the components with the second type of tuple, and etc.

Vertical Automorphism

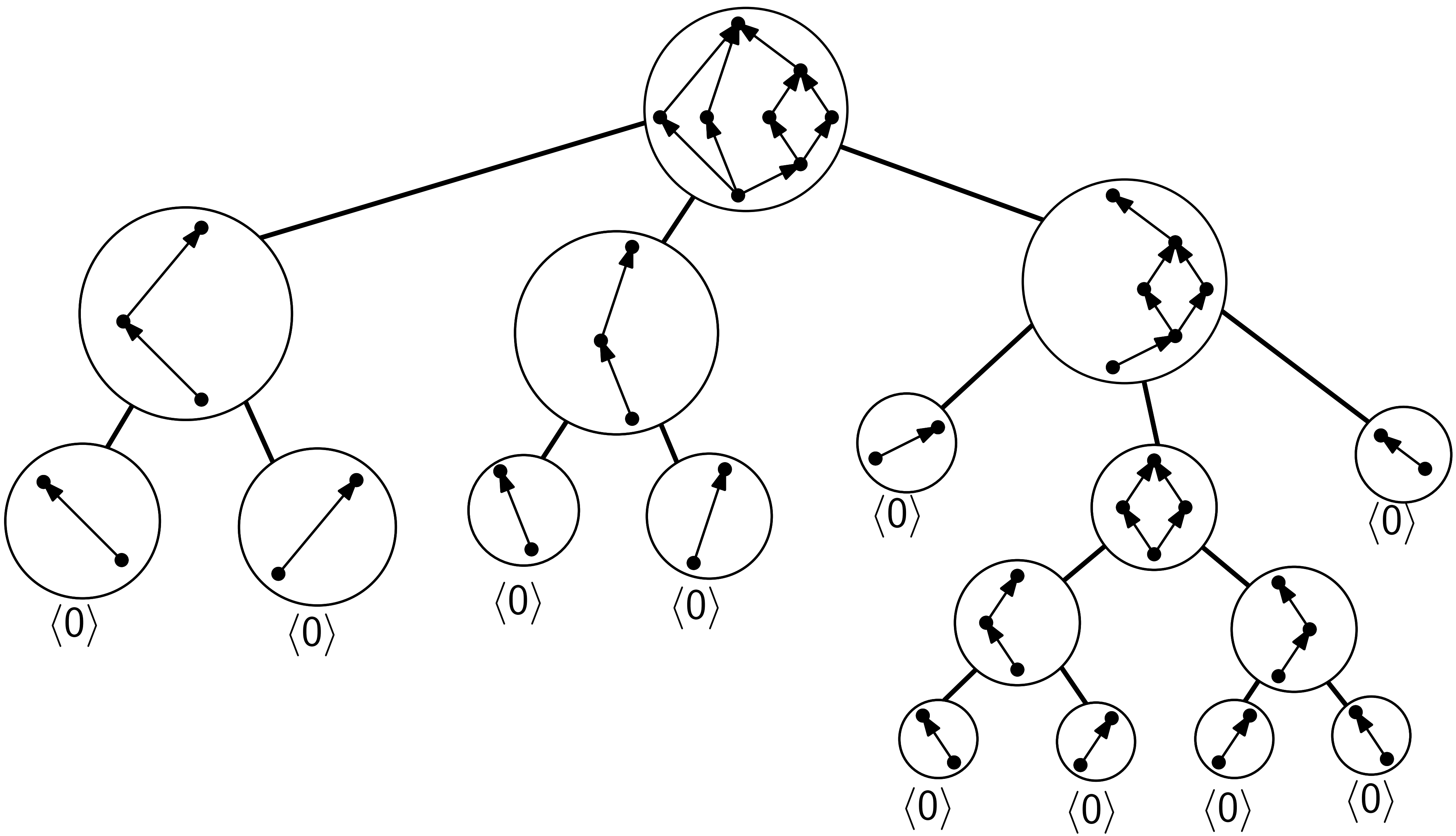
Algorithm constructing a Canonical Labeling

- Set $tuple(G_i) = \langle 0 \rangle$ for all Q-nodes G_i of G.
- For each $t = \max \text{depth}(G), \ldots, 0$
  - For each S- or P-node G' at depth t with children $G_1, \ldots, G_k$ set $tuple(G') = \langle code(G_1), \ldots, code(G_k) \rangle$. If G' is a P-node, sort $tuple(G')$ in non-decreasing order.
  - Sort all the nodes at depth t lexicografically according to tuples.
  - For each component G' at depth t, compute $code(G')$ as follows. Assign the integer 1 to those components represented by the first distinct tuple, assign 2 to the components with the second type of tuple, and etc.

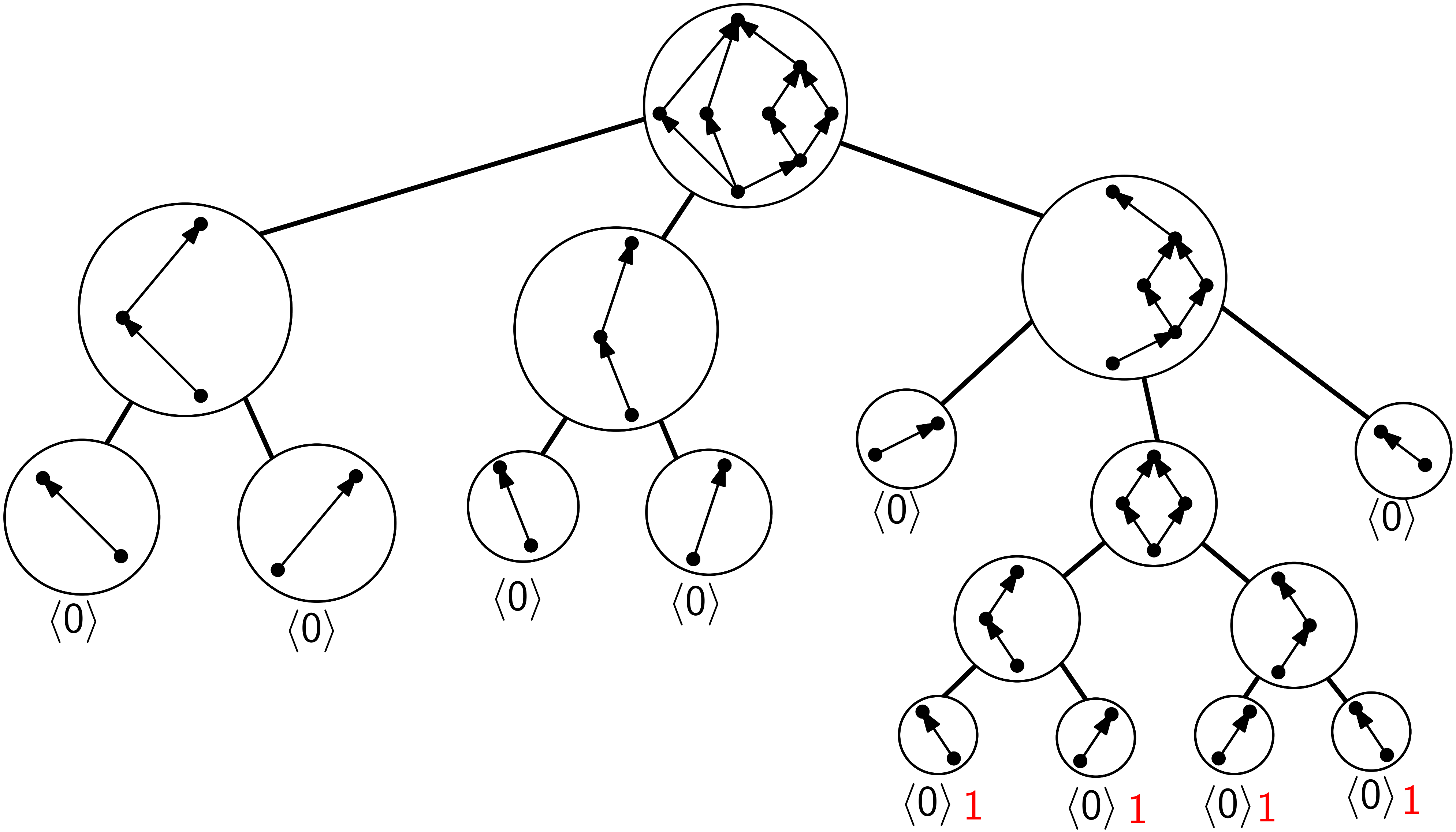
Lemma

Two nodes u and v at the same depth of the decomposition tree of G represent isomorphic subgraphs of G iff $code(u) = code(v)$.

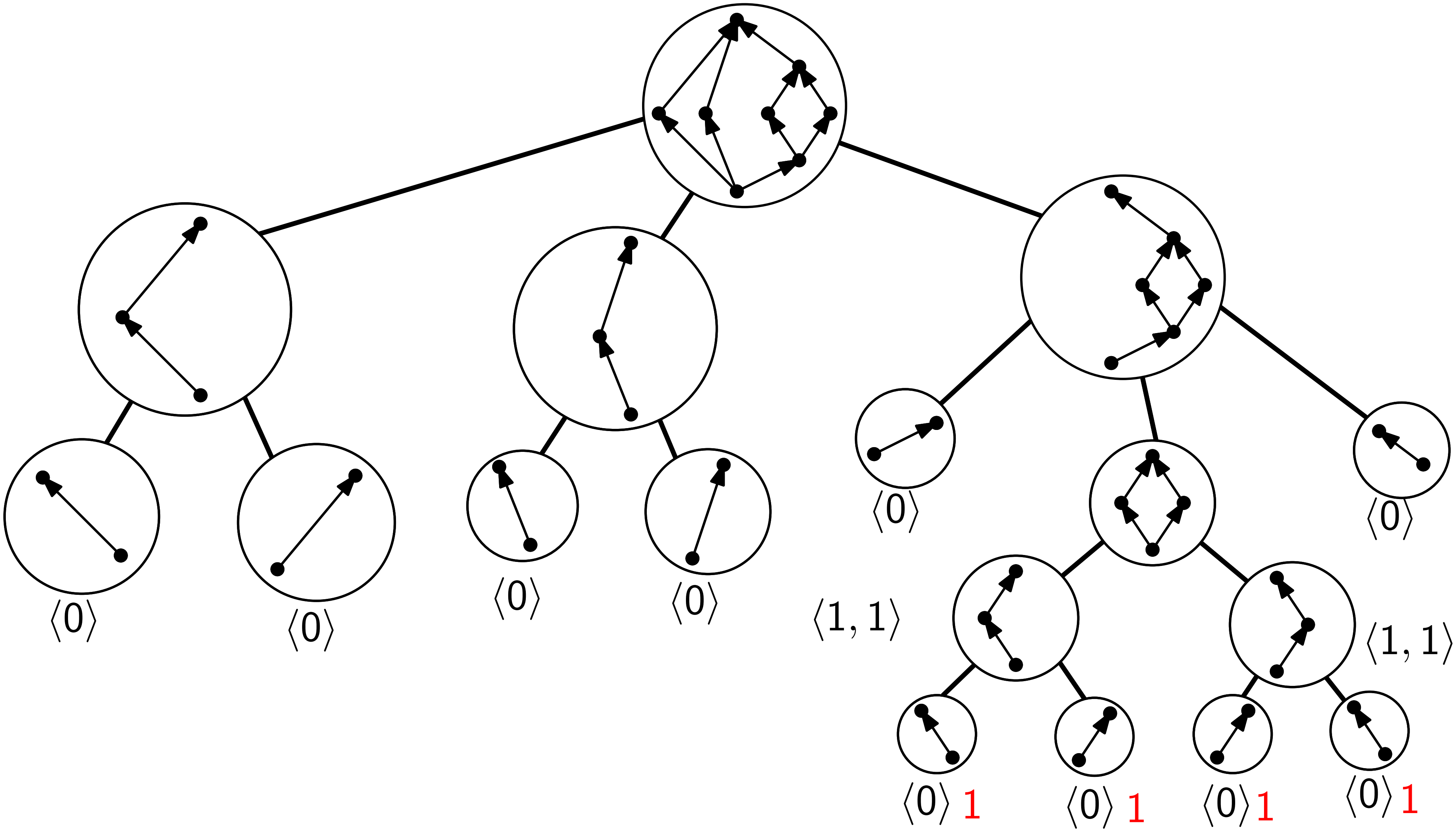
Vertical Automorphism



Vertical Automorphism

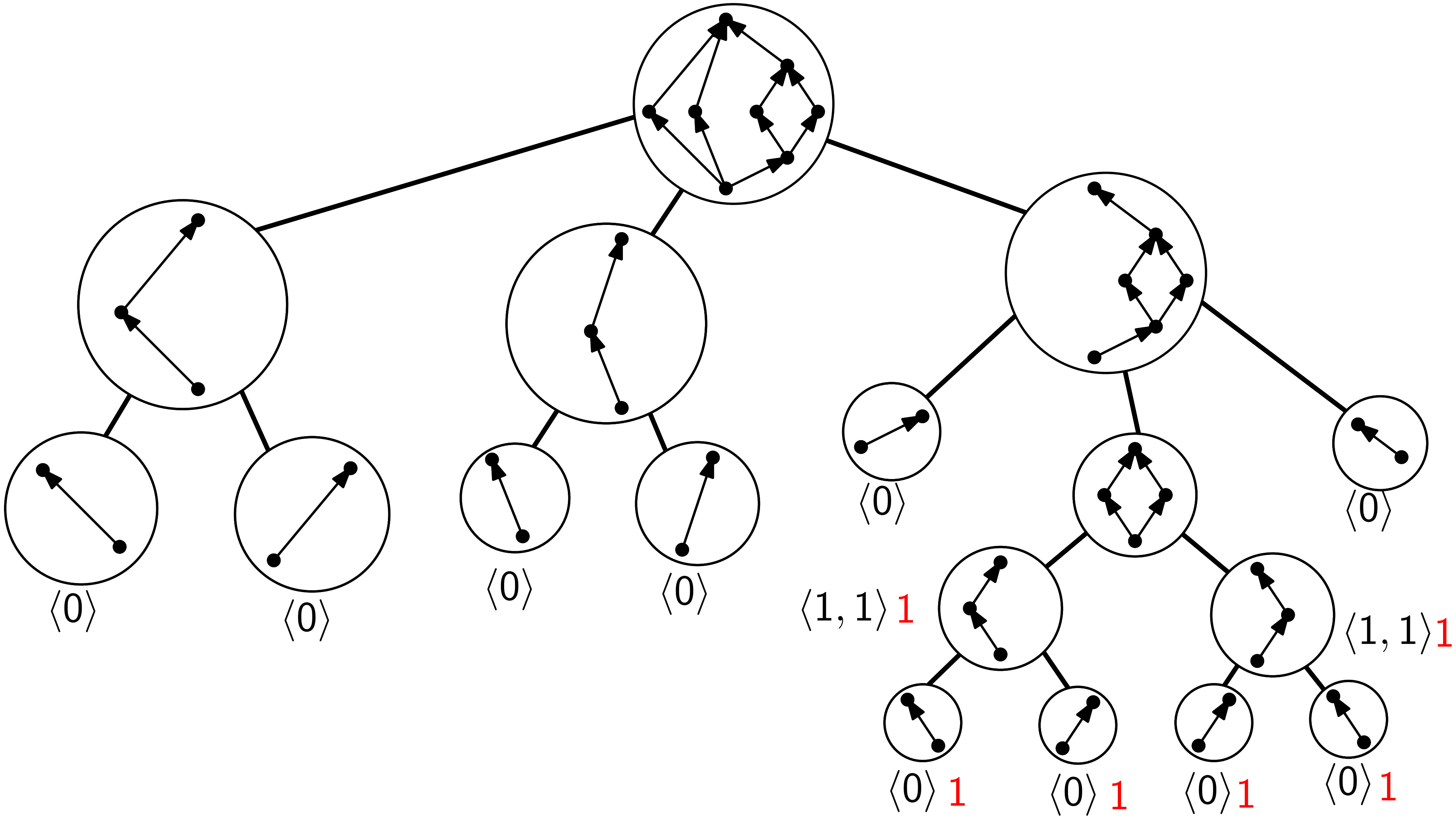


Vertical Automorphism

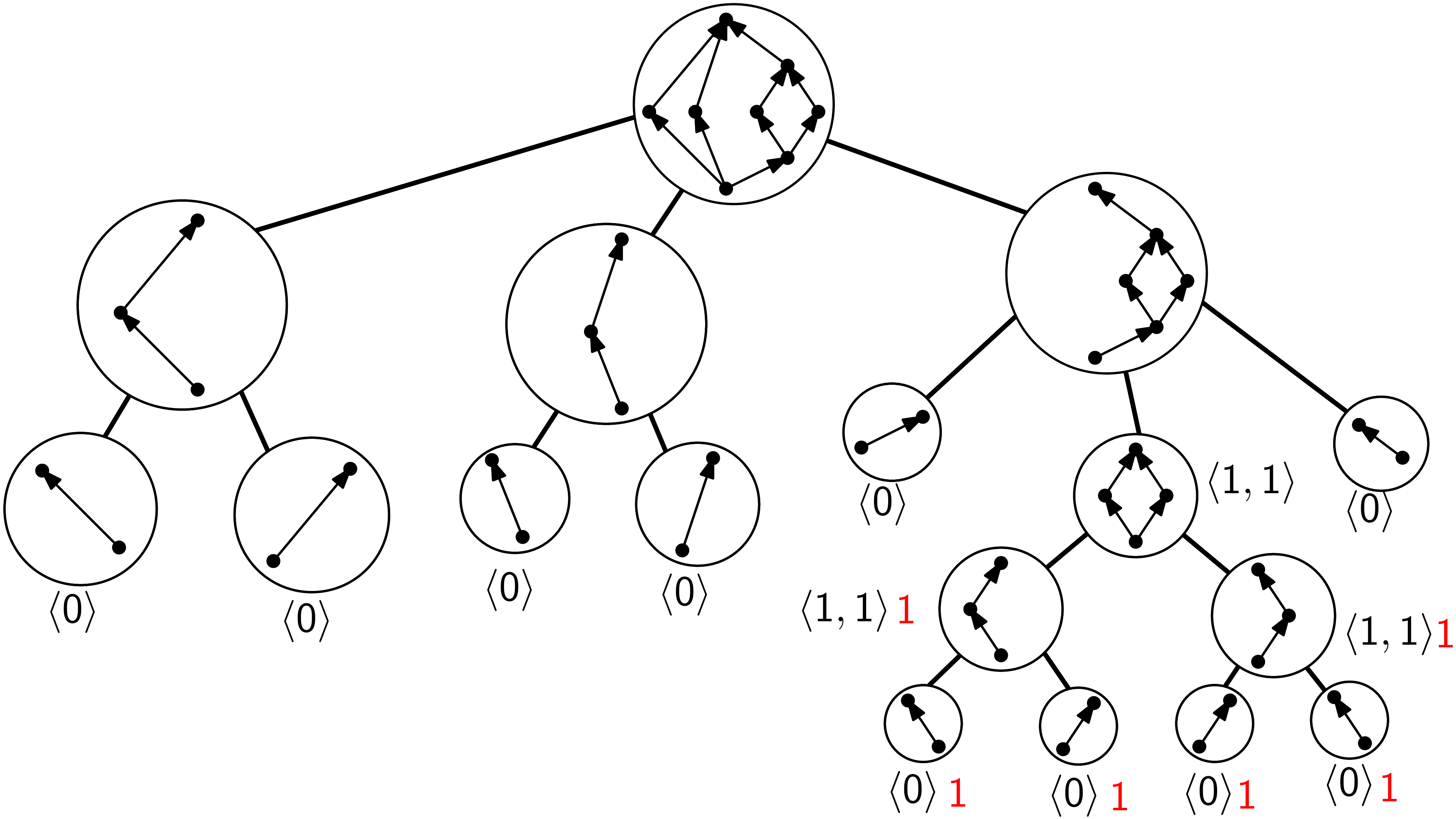


Institut für Theoretische Informatik
Lehrstuhl Algorithmik I

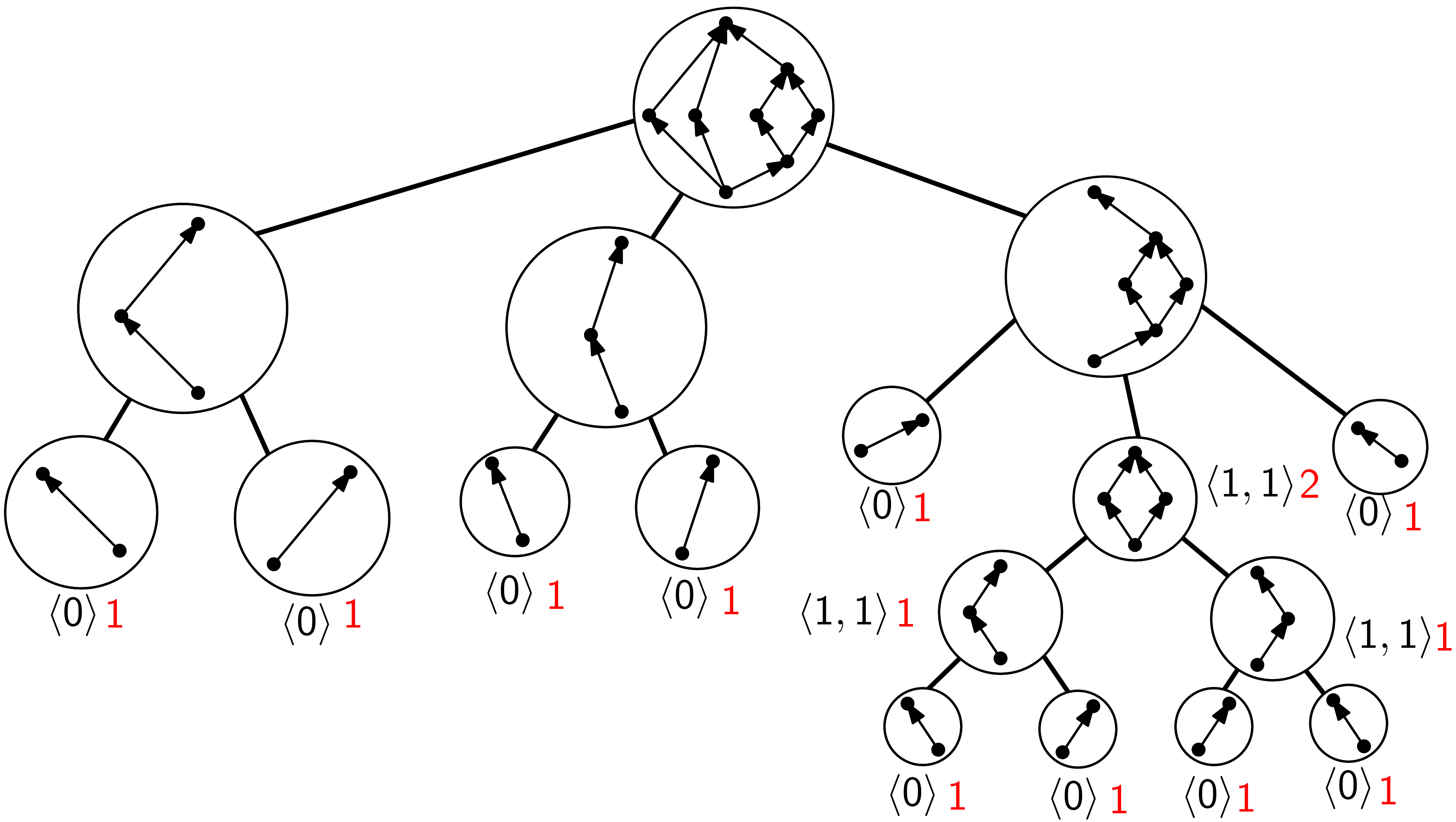
Vertical Automorphism



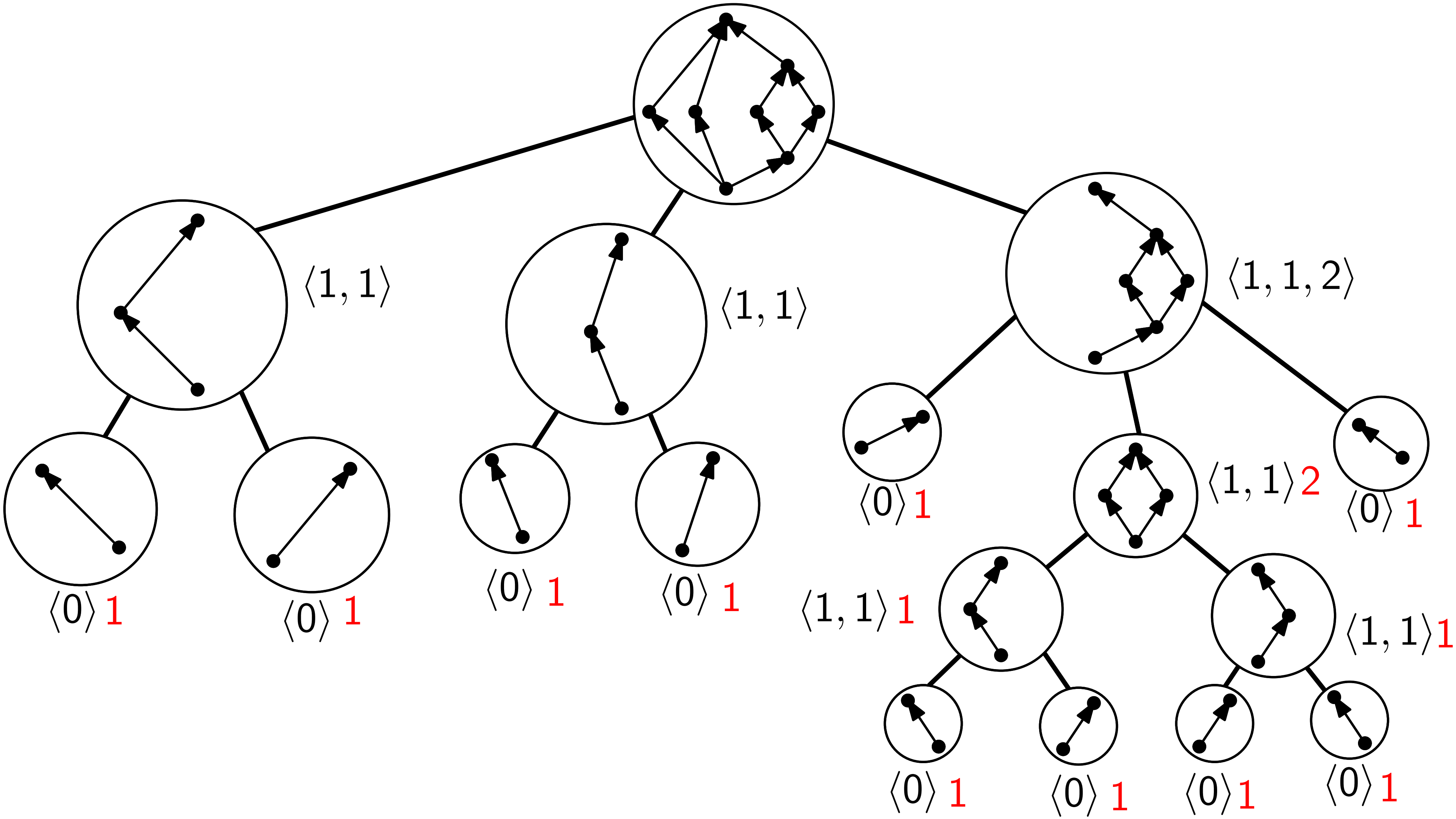
Vertical Automorphism



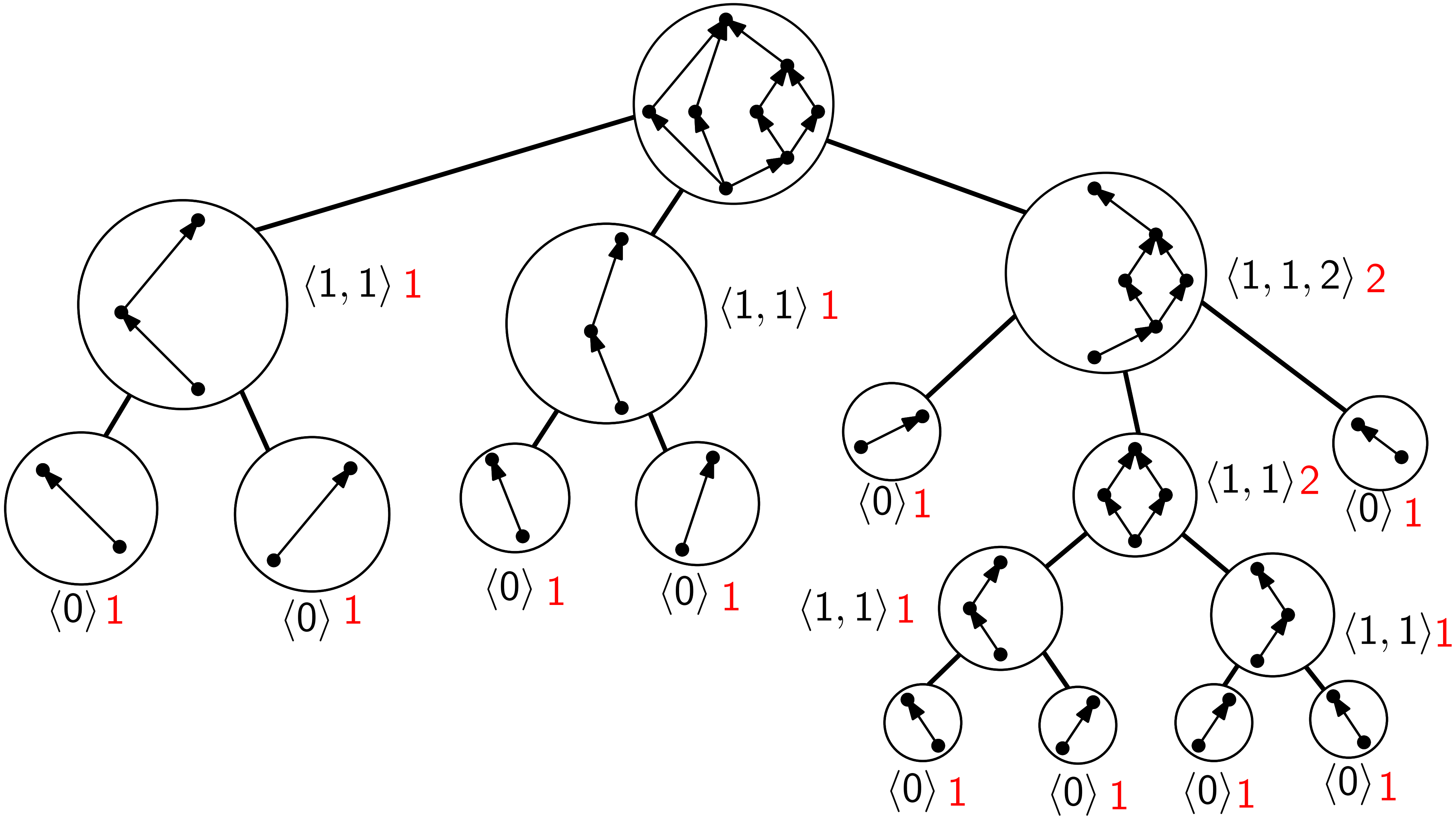
Vertical Automorphism



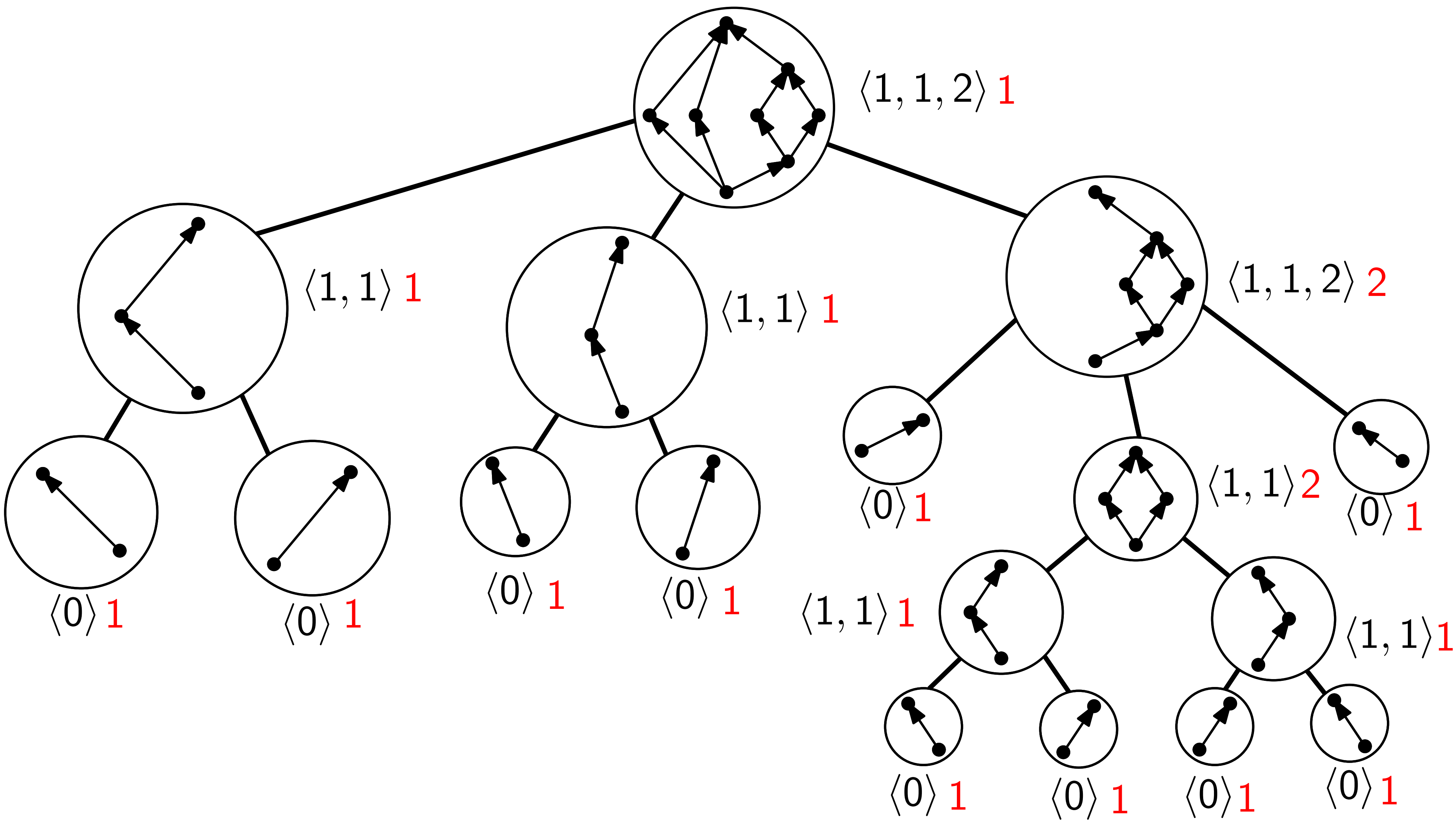
Vertical Automorphism



Vertical Automorphism



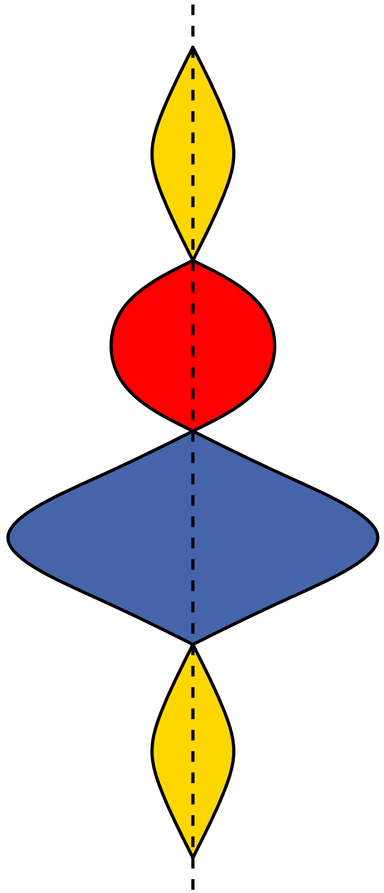
Vertical Automorphism



Institut für Theoretische Informatik
Lehrstuhl Algorithmik I

Vertical Automorphism

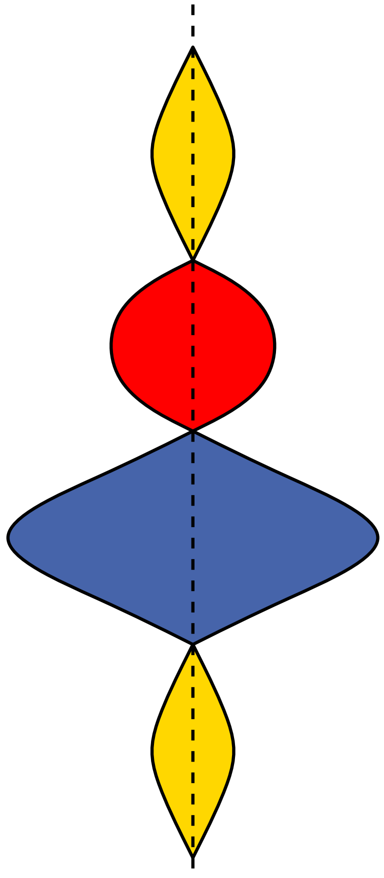
- Let G be composed out of $G_1 \dots G_n$ through series or parallel composition, $tuple(G)$ contains the codes of $G_1, \dots, G_n$.
- How can we use $tuple(G)$ do decide whether G has a vertical automorphism?



G is an S-node

Vertical Automorphism

- Let G be composed out of $G_1 \dots G_n$ through series or parallel composition, $tuple(G)$ contains the codes of $G_1, \dots, G_n$.
- How can we use $tuple(G)$ do decide whether G has a vertical automorphism?



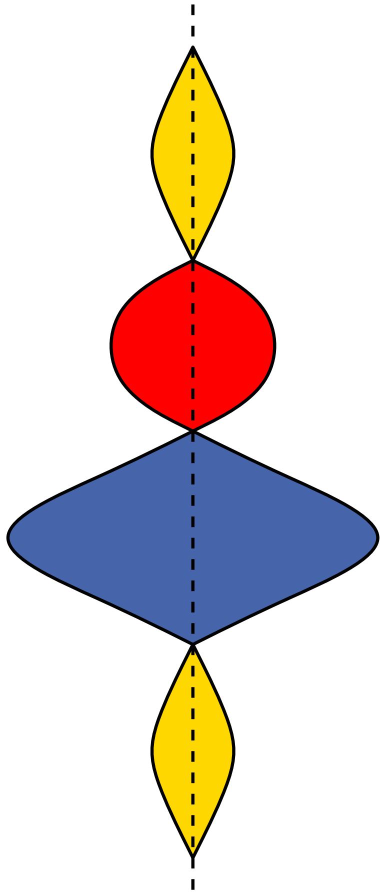
G is an S-node

Lemma (Hong, Eades, Lee '00)

If G is an S-node, then G has a vertical automorphism iff each of $G_1, \dots, G_k$ has a vertical automorphism.

Vertical Automorphism

- Let G be composed out of $G_1 \ldots G_n$ through series or parallel composition, $tuple(G)$ contains the codes of $G_1, \ldots, G_n$.
- How can we use $tuple(G)$ do decide whether G has a vertical automorphism?



G is an S-node

Lemma (Hong, Eades, Lee '00)

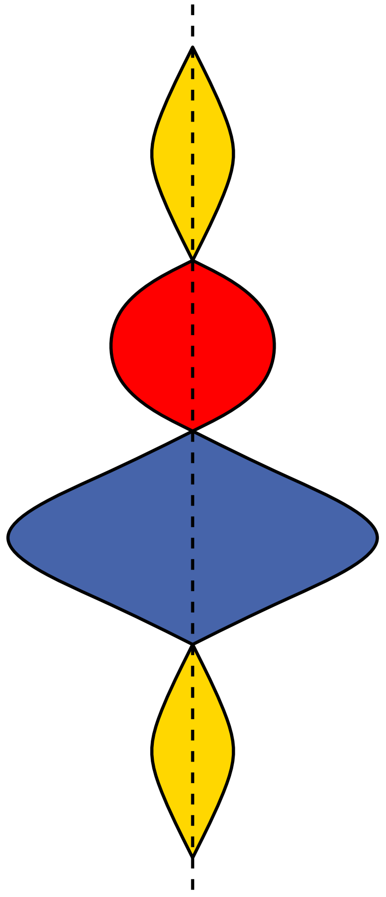
If G is an S-node, then G has a vertical automorphism iff each of $G_1, \ldots, G_k$ has a vertical automorphism.

Proof:

- Assume G has a vertical automorphism α

Vertical Automorphism

- Let G be composed out of $G_1 \dots G_n$ through series or parallel composition, $tuple(G)$ contains the codes of $G_1, \dots, G_n$.
- How can we use $tuple(G)$ do decide whether G has a vertical automorphism?



G is an S-node

Lemma (Hong, Eades, Lee '00)

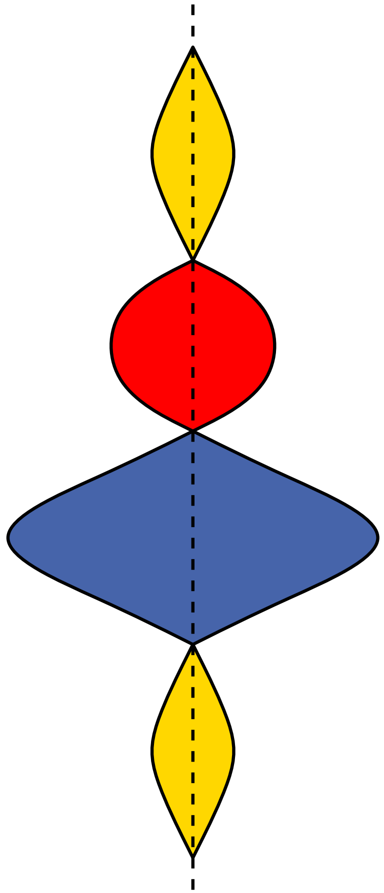
If G is an S-node, then G has a vertical automorphism iff each of $G_1, \dots, G_k$ has a vertical automorphism.

Proof:

- Assume G has a vertical automorphism α
- Then α "fixes" all the components

Vertical Automorphism

- Let G be composed out of $G_1 \dots G_n$ through series or parallel composition, $tuple(G)$ contains the codes of $G_1, \dots, G_n$.
- How can we use $tuple(G)$ do decide whether G has a vertical automorphism?



G is an S-node

Lemma (Hong, Eades, Lee '00)

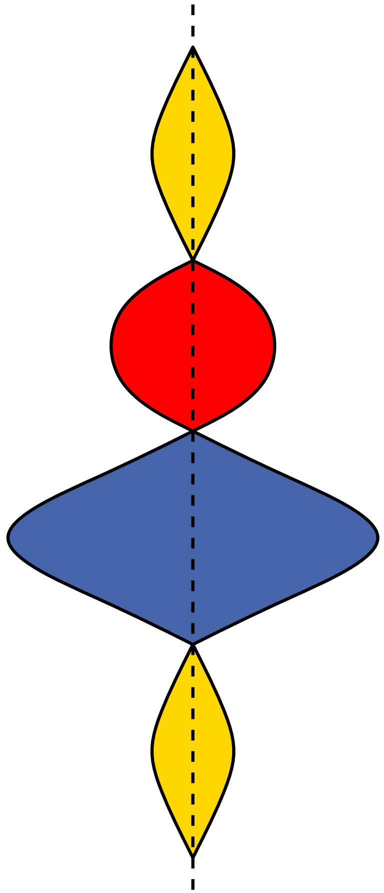
If G is an S-node, then G has a vertical automorphism iff each of $G_1, \dots, G_k$ has a vertical automorphism.

Proof:

- Assume G has a vertical automorphism α
- Then α "fixes" all the components
- Therefore each of the series components has a vertical automorhism

Vertical Automorphism

- Let G be composed out of $G_1 \ldots G_n$ through series or parallel composition, $tuple(G)$ contains the codes of $G_1, \ldots, G_n$.
- How can we use $tuple(G)$ do decide whether G has a vertical automorphism?



G is an S-node

Lemma (Hong, Eades, Lee '00)

If G is an S-node, then G has a vertical automorphism iff each of $G_1, \ldots, G_k$ has a vertical automorphism.

Proof:

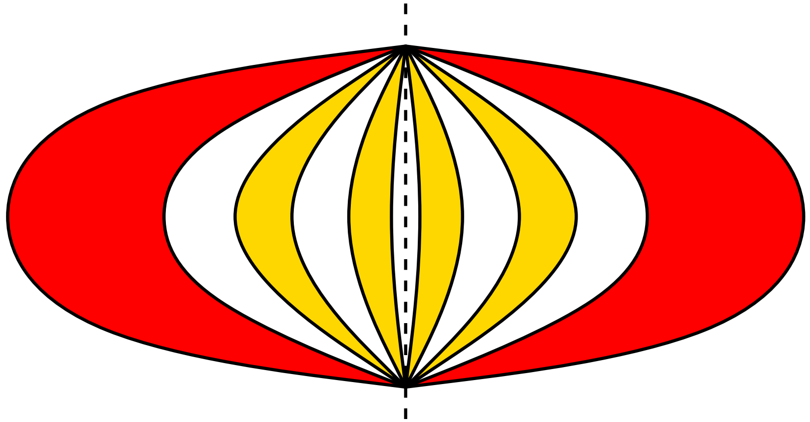
- Assume G has a vertical automorphism α
- Then α "fixes" all the components
- Therefore each of the series components has a vertical automorhism
- If each of $G_1, \ldots, G_n$ has a vertical isomorphism, arrange them as in Figure.

Vertical Automorphism

Lemma (Hong, Eades, Lee '00)

If G is a P-node, consider a partition of $\mathcal{C}_j = \{G_i : 1 \leq i \leq k, code(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.

- If $\forall j$, $|\mathcal{C}_j|$ are even $\Rightarrow$ has a vertical automorphism.
- If there exists a unique j, such that $|\mathcal{C}_j|$ is odd $\Rightarrow G$ has a vertical automorphism iff graphs of $\mathcal{C}_j$ have a vertical automorphism.
- If there exists $|\mathcal{C}_i|, |\mathcal{C}_j|$ with $i \neq j$, both odd $\Rightarrow G$ does not have a vertical automorphism.



Proof:

- Arrange components as in Figure.

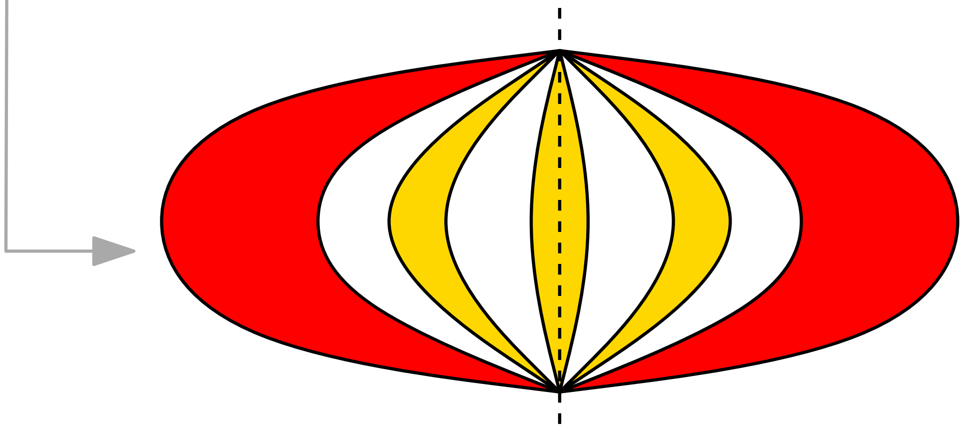
G is P-node, $tuple(G) = < \underbrace{1 \ldots 1}_{even}, \underbrace{2 \ldots 2}_{even}, \cdots >$

Vertical Automorphism

Lemma (Hong, Eades, Lee '00)

If G is a P-node, consider a partition of $\mathcal{C}_j = \{G_i : 1 \leq i \leq k, code(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.

- If $\forall j$, $|\mathcal{C}_j|$ are even $\Rightarrow$ has a vertical automorphism.
- If there exists a unique j, such that $|\mathcal{C}_j|$ is odd $\Rightarrow G$ has a vertical automorphism iff graphs of $\mathcal{C}_j$ have a vertical automorphism.
- If there exists $|\mathcal{C}_i|, |\mathcal{C}_j|$ with $i \neq j$, both odd $\Rightarrow G$ does not have a vertical automorphism.



$$tuple(G) = < \underbrace{1 \ldots 1}_{odd}, \underbrace{2 \ldots 2}_{even}, \underbrace{3 \ldots 3}_{even}, \cdots >$$

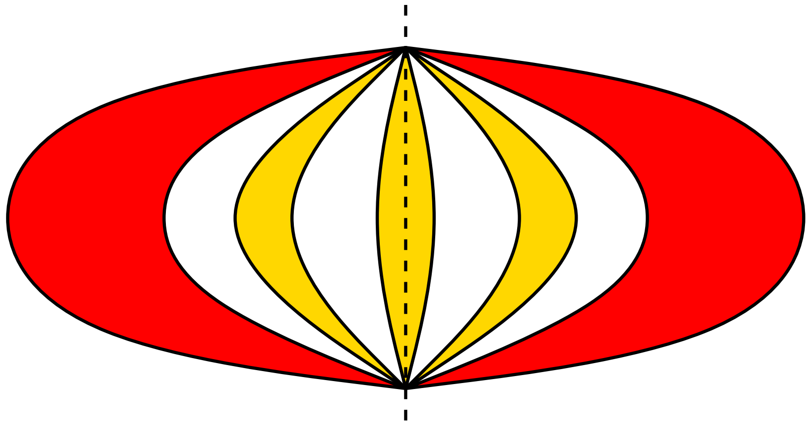
Proof:

Vertical Automorphism

Lemma (Hong, Eades, Lee '00)

If G is a P-node, consider a partition of $\mathcal{C}_j = \{G_i : 1 \leq i \leq k, code(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.

- If $\forall j$, $|\mathcal{C}_j|$ are even $\Rightarrow$ has a vertical automorphism.
- If there exists a unique j, such that $|\mathcal{C}_j|$ is odd $\Rightarrow$ G has a vertical automorphism iff graphs of $\mathcal{C}_j$ have a vertical automorphism.
- If there exists $|\mathcal{C}_i|, |\mathcal{C}_j|$ with $i \neq j$, both odd $\Rightarrow$ G does not have a vertical automorphism.



$$tuple(G) =< \underbrace{1 \ldots 1}_{odd}, \underbrace{2 \ldots 2}_{even}, \underbrace{3 \ldots 3}_{even}, \cdots >$$

Proof:

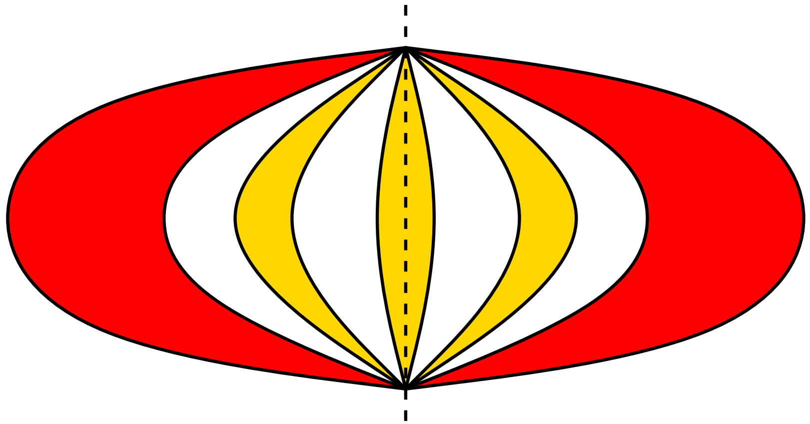
- Any vertical automorphism "fixes" a member of $\mathcal{C}_j$, therefore it has a vertical automorphism.

Vertical Automorphism

Lemma (Hong, Eades, Lee ’00)

If G is a P-node, consider a partition of $\mathcal{C}_j = \{G_i : 1 \leq i \leq k, code(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.

- If $\forall j$, $|\mathcal{C}_j|$ are even $\Rightarrow$ has a vertical automorphism.
- If there exists a unique j, such that $|\mathcal{C}_j|$ is odd $\Rightarrow$ G has a vertical automorphism iff graphs of $\mathcal{C}_j$ have a vertical automorphism.
- If there exists $|\mathcal{C}_i|, |\mathcal{C}_j|$ with $i \neq j$, both odd $\Rightarrow$ G does not have a vertical automorphism.



$$tuple(G) =< \underbrace{1 \ldots 1}_{odd}, \underbrace{2 \ldots 2}_{even}, \underbrace{3 \ldots 3}_{even}, \cdots >$$

Proof:

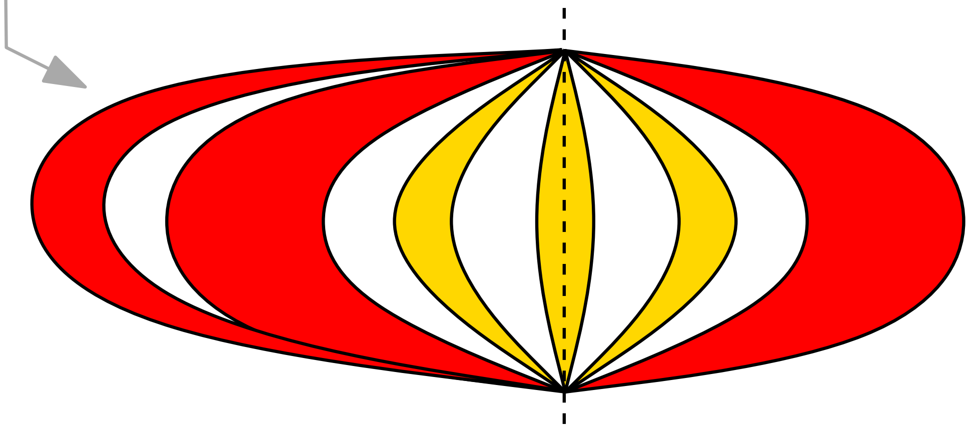
- Any vertical automorphism “fixes” a member of $\mathcal{C}_j$, therefore it has a vertical automorphism.
- Conversly, arrange as in figure.

Vertical Automorphism

Lemma (Hong, Eades, Lee ’00)

If G is a P-node, consider a partition of $\mathcal{C}_j = \{G_i : 1 \leq i \leq k, code(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.

- If $\forall j$, $|\mathcal{C}_j|$ are even $\Rightarrow$ has a vertical automorphism.
- If there exists a unique j, such that $|\mathcal{C}_j|$ is odd $\Rightarrow$ G has a vertical automorphism iff graphs of $\mathcal{C}_j$ have a vertical automorphism.
- If there exists $|\mathcal{C}_i|, |\mathcal{C}_j|$ with $i \neq j$, both odd $\Rightarrow$ G does not have a vertical automorphism.



$$tuple(G) =< \underbrace{1 \ldots 1}_{odd}, \underbrace{2 \ldots 2}_{odd}, \underbrace{3 \ldots 3}_{even}, \cdots >$$

Proof:

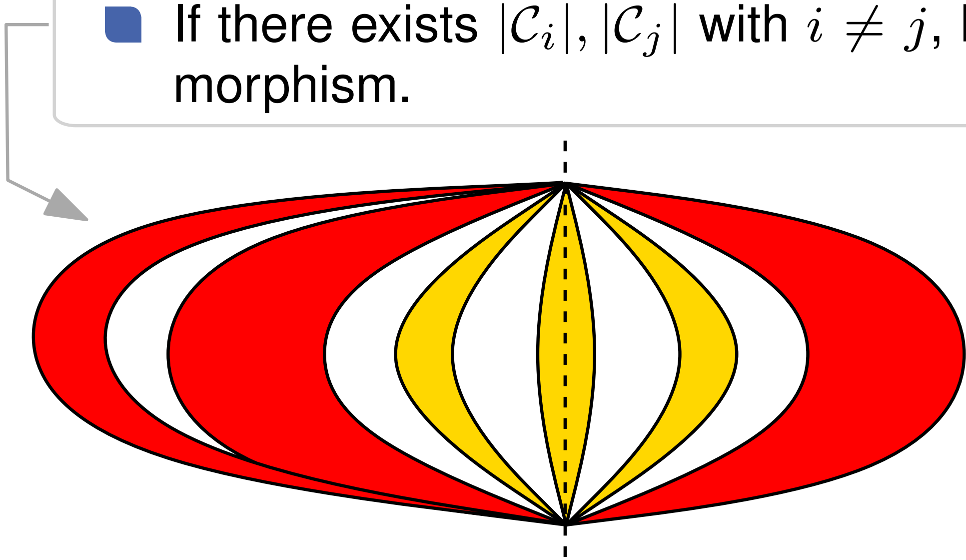
Vertical Automorphism

KIT
Karlsruhe Institute of Technology

Lemma (Hong, Eades, Lee '00)

If G is a P-node, consider a partition of $\mathcal{C}_j = \{G_i : 1 \leq i \leq k, code(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.

- If $\forall j$, $|\mathcal{C}_j|$ are even $\Rightarrow$ has a vertical automorphism.
- If there exists a unique j, such that $|\mathcal{C}_j|$ is odd $\Rightarrow$ G has a vertical automorphism iff graphs of $\mathcal{C}_j$ have a vertical automorphism.
- If there exists $|\mathcal{C}_i|, |\mathcal{C}_j|$ with $i \neq j$, both odd $\Rightarrow$ G does not have a vertical automorphism.



$$tuple(G) =< \underbrace{1 \ldots 1}_{odd}, \underbrace{2 \ldots 2}_{odd}, \underbrace{3 \ldots 3}_{even}, \cdots >$$

Proof:

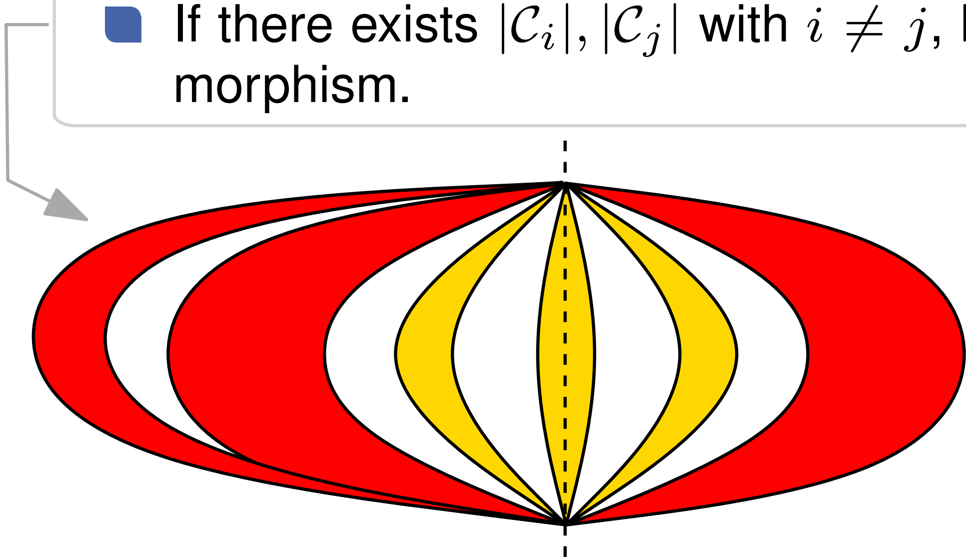
- Any vertical automorphism has to "fix" two distinct components.

Vertical Automorphism

Lemma (Hong, Eades, Lee '00)

If G is a P-node, consider a partition of $\mathcal{C}_j = \{G_i : 1 \leq i \leq k, code(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.

- If $\forall j$, $|\mathcal{C}_j|$ are even $\Rightarrow$ has a vertical automorphism.
- If there exists a unique j, such that $|\mathcal{C}_j|$ is odd $\Rightarrow$ G has a vertical automorphism iff graphs of $\mathcal{C}_j$ have a vertical automorphism.
- If there exists $|\mathcal{C}_i|, |\mathcal{C}_j|$ with $i \neq j$, both odd $\Rightarrow$ G does not have a vertical automorphism.



$$tuple(G) = < \underbrace{1 \ldots 1}_{odd}, \underbrace{2 \ldots 2}_{odd}, \underbrace{3 \ldots 3}_{even}, \cdots >$$

Proof:

- Any vertical automorphism has to "fix" two distinct components.
- In both components we can find a path on which some vertices are aligned on the axis. Contradicts planarity.

Vertical Automorphism

Theorem (Hong, Eades, Lee '00)

Given a decomposition tree of a series-parallel graph and its canonical labeling. Let G be a component which consists from $G_1, \ldots, G_k$ through series or parallel composition.

- If G is an S-node, then G has a vertical automorphism iff each of $G_1, \ldots, G_k$ has a vertical automorphism.

Vertical Automorphism

Theorem (Hong, Eades, Lee ’00)

Given a decomposition tree of a series-parallel graph and its canonical labeling. Let G be a component which consists from $G_1, \ldots, G_k$ through series or parallel composition.

- If G is an S-node, then G has a vertical automorphism iff each of $G_1, \ldots, G_k$ has a vertical automorphism.
- If G is a P-node, consider a partition of $\mathcal{C}_j = \{G_i : 1 \leq i \leq k, code(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.
  - If $\forall j$, $|\mathcal{C}_j|$ are even $\Rightarrow$ has a vertical automorphism.

Vertical Automorphism

Theorem (Hong, Eades, Lee '00)

Given a decomposition tree of a series-parallel graph and its canonical labeling. Let G be a component which consists from $G_1, \ldots, G_k$ through series or parallel composition.

- If G is an S-node, then G has a vertical automorphism iff each of $G_1, \ldots, G_k$ has a vertical automorphism.
- If G is a P-node, consider a partition of $\mathcal{C}_j = \{G_i : 1 \leq i \leq k, code(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.
  - If $\forall j$, $|\mathcal{C}_j|$ are even $\Rightarrow$ has a vertical automorphism.
  - If there exists a unique j, such that $|\mathcal{C}_j|$ is odd $\Rightarrow$ G has a vertical automorphism iff graphs of $\mathcal{C}_j$ have a vertical automorphism.

Vertical Automorphism

Theorem (Hong, Eades, Lee '00)

Given a decomposition tree of a series-parallel graph and its canonical labeling. Let G be a component which consists from $G_1, \ldots, G_k$ through series or parallel composition.

- If G is an S-node, then G has a vertical automorphism iff each of $G_1, \ldots, G_k$ has a vertical automorphism.
- If G is a P-node, consider a partition of $\mathcal{C}_j = \{G_i : 1 \leq i \leq k, code(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.
  - If $\forall j$, $|\mathcal{C}_j|$ are even $\Rightarrow$ has a vertical automorphism.
  - If there exists a unique j, such that $|\mathcal{C}_j|$ is odd $\Rightarrow$ G has a vertical automorphism iff graphs of $\mathcal{C}_j$ have a vertical automorphism.
  - If there exists $|\mathcal{C}_i|, |\mathcal{C}_j|$ with $i \neq j$, both odd $\Rightarrow$ G does not have a vertical automorphism.