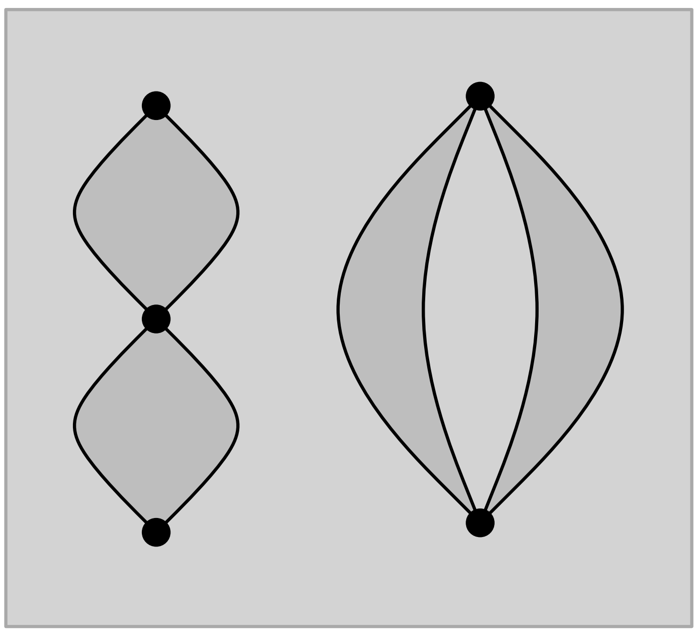
Karlsruhe Institute of Technology

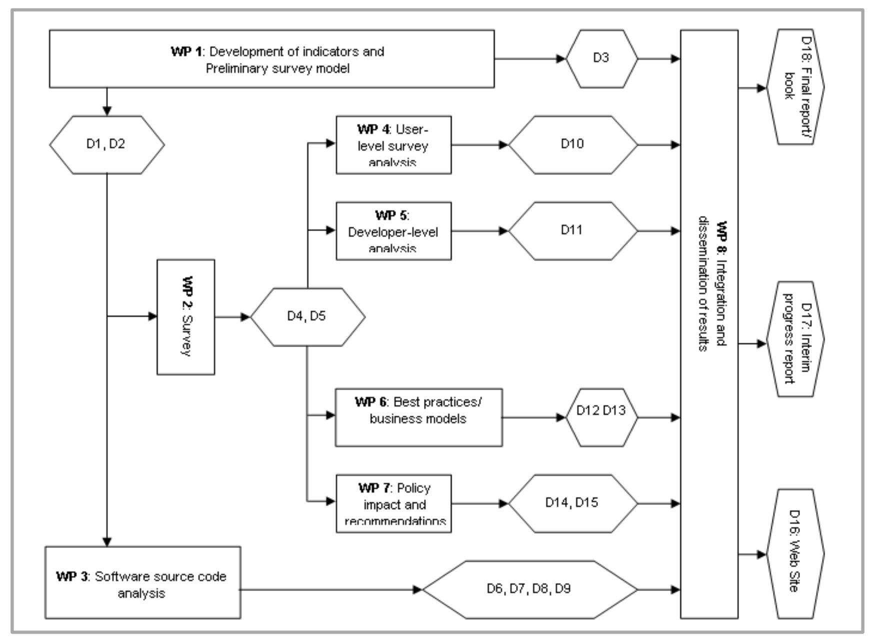
Algorithms for graph visualization

Divide and Conquer - Series-Parallel Graphs

WINTER SEMESTER 2017/2018

Tamara Mchedlidze





Overview

Series - parallel graph. Definition and Decomposition.

Algorithm for upward straight-line drawing.

Lower bound on the area.

Symmetry display for series-parallel graphs.

Overview

Series - parallel graph. Definition and Decomposition.

Algorithm for upward straight-line drawing.

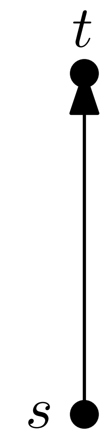
Lower bound on the area.

Symmetry display for series-parallel graphs.

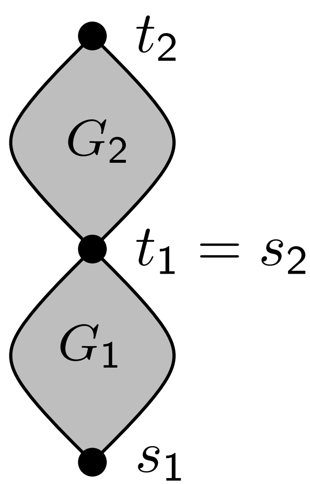
Series-parallel Graphs

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

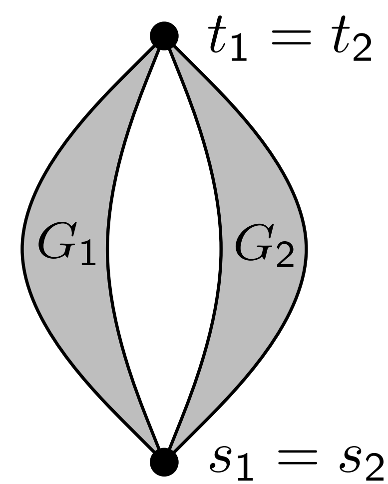
- It contains a single edge (s, t) (s-source, t-sink)
- It consists of two series-parallel graphs G_1, G_2 with sources s_1, s_2 and sinks t_1, t_2 which are combined using one of the following rules:



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

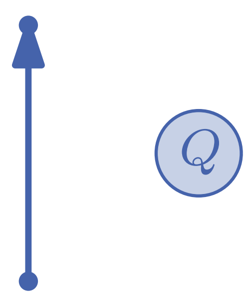


A **decomposition tree** of G, which is a binary tree T with nodes of three types: S,P and Q-type.

Series-parallel Graphs. Decomposition Tree.

A **decomposition tree** of G, which is a binary tree T with nodes of three types: S,P and Q-type.

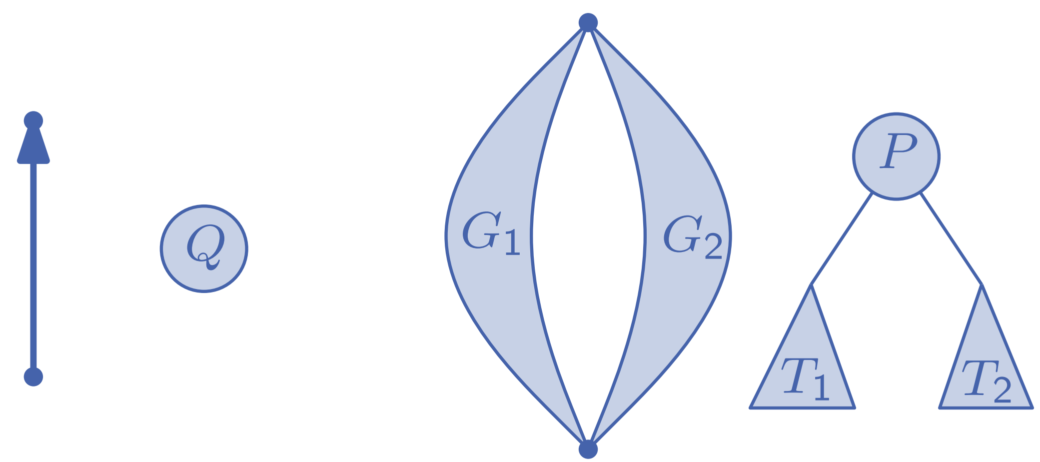
- If G is a single edge, then the corresponding node is Q-node



Series-parallel Graphs. Decomposition Tree.

A **decomposition tree** of G, which is a binary tree T with nodes of three types: S,P and Q-type.

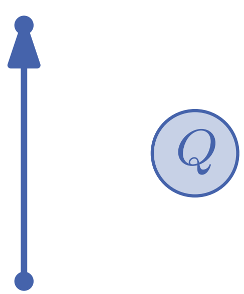
- If G is a single edge, then the corresponding node is Q-node
- If G is a parallel composition of G_1 (with tree T_1) and G_2 (with tree T_2), then the root of T is P-node and T_1 is its left subtree, T_2 is its right subtree

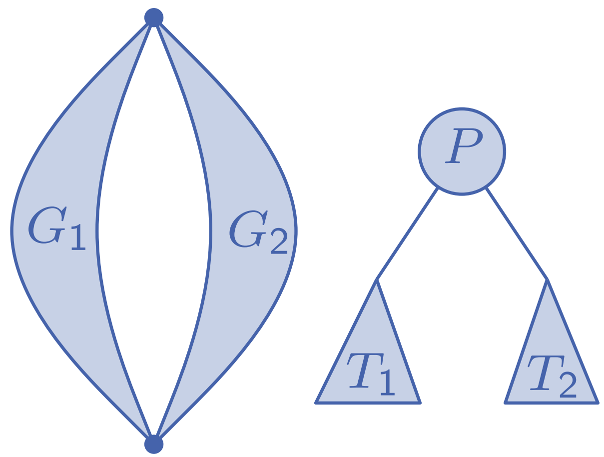


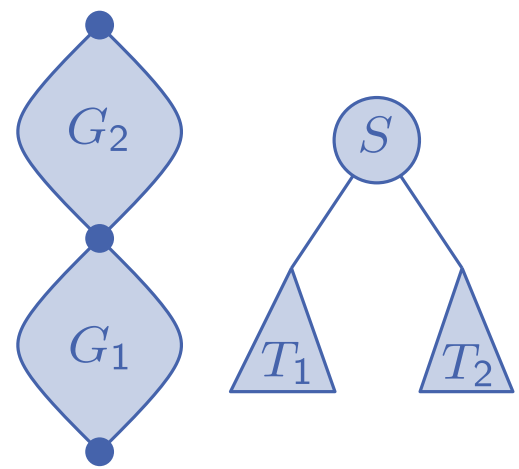
Series-parallel Graphs. Decomposition Tree.

A **decomposition tree** of G, which is a binary tree T with nodes of three types: S,P and Q-type.

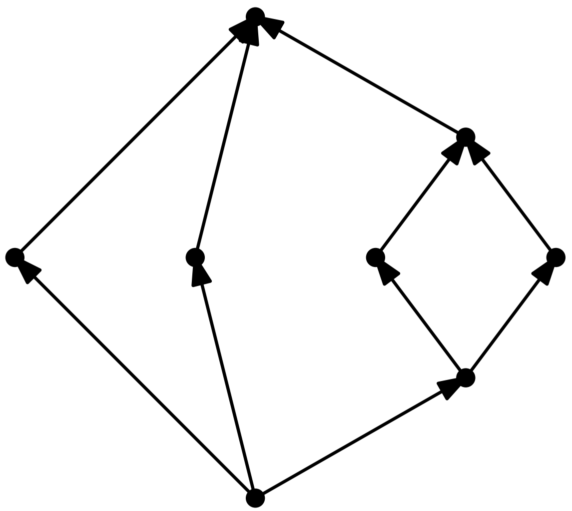
- If G is a single edge, then the corresponding node is Q-node
- If G is a parallel composition of G_1 (with tree T_1) and G_2 (with tree T_2), then the root of T is P-node and T_1 is its left subtree, T_2 is its right subtree
- If G is a series composition of G_1 (with tree T_1) and G_2 (with tree T_2),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





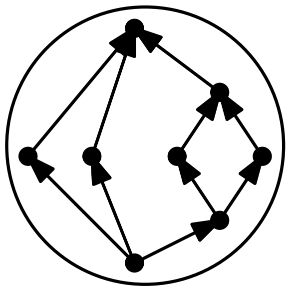


Series-parallel Graphs. Decomposition Example.

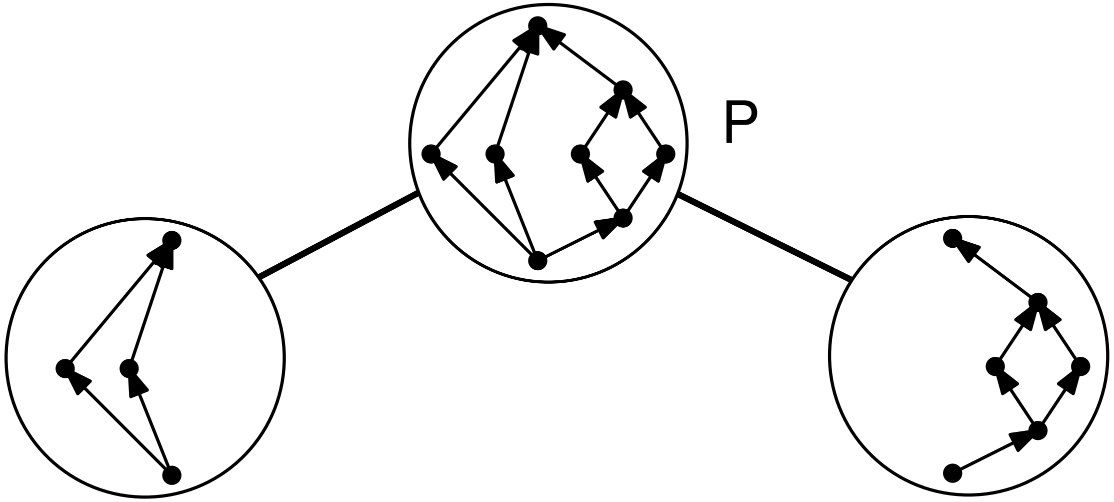


6 - 1

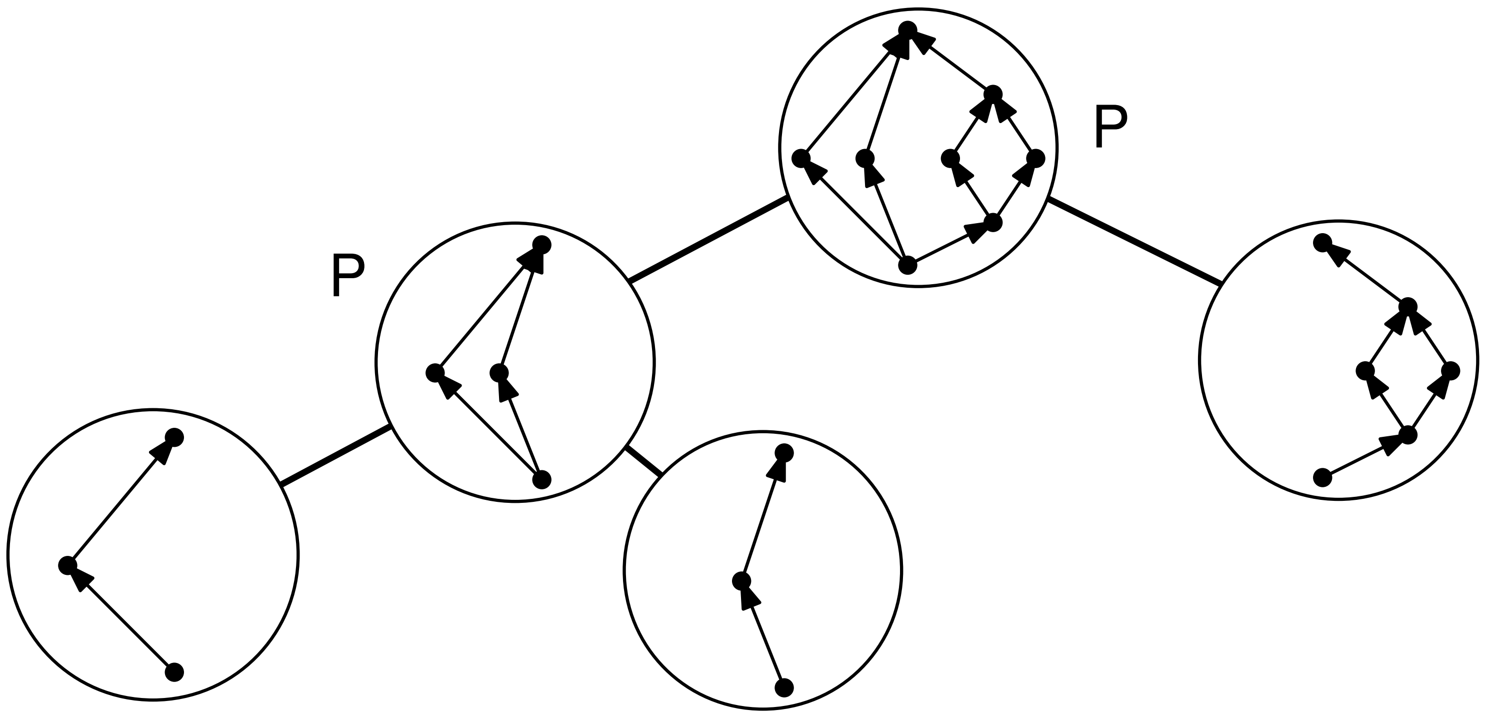
Series-parallel Graphs. Decomposition Example.



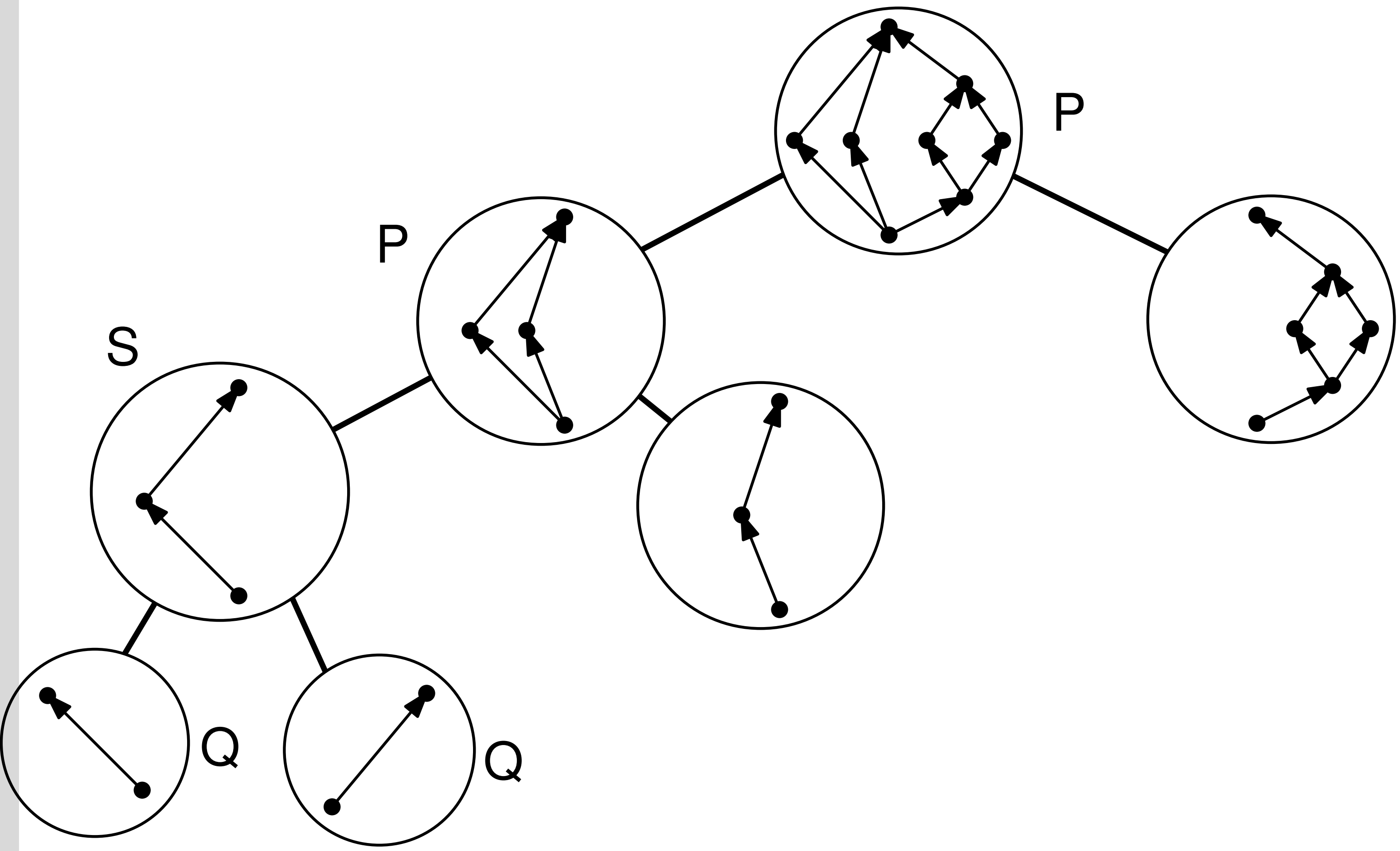
Series-parallel Graphs. Decomposition Example.



Series-parallel Graphs. Decomposition Example.

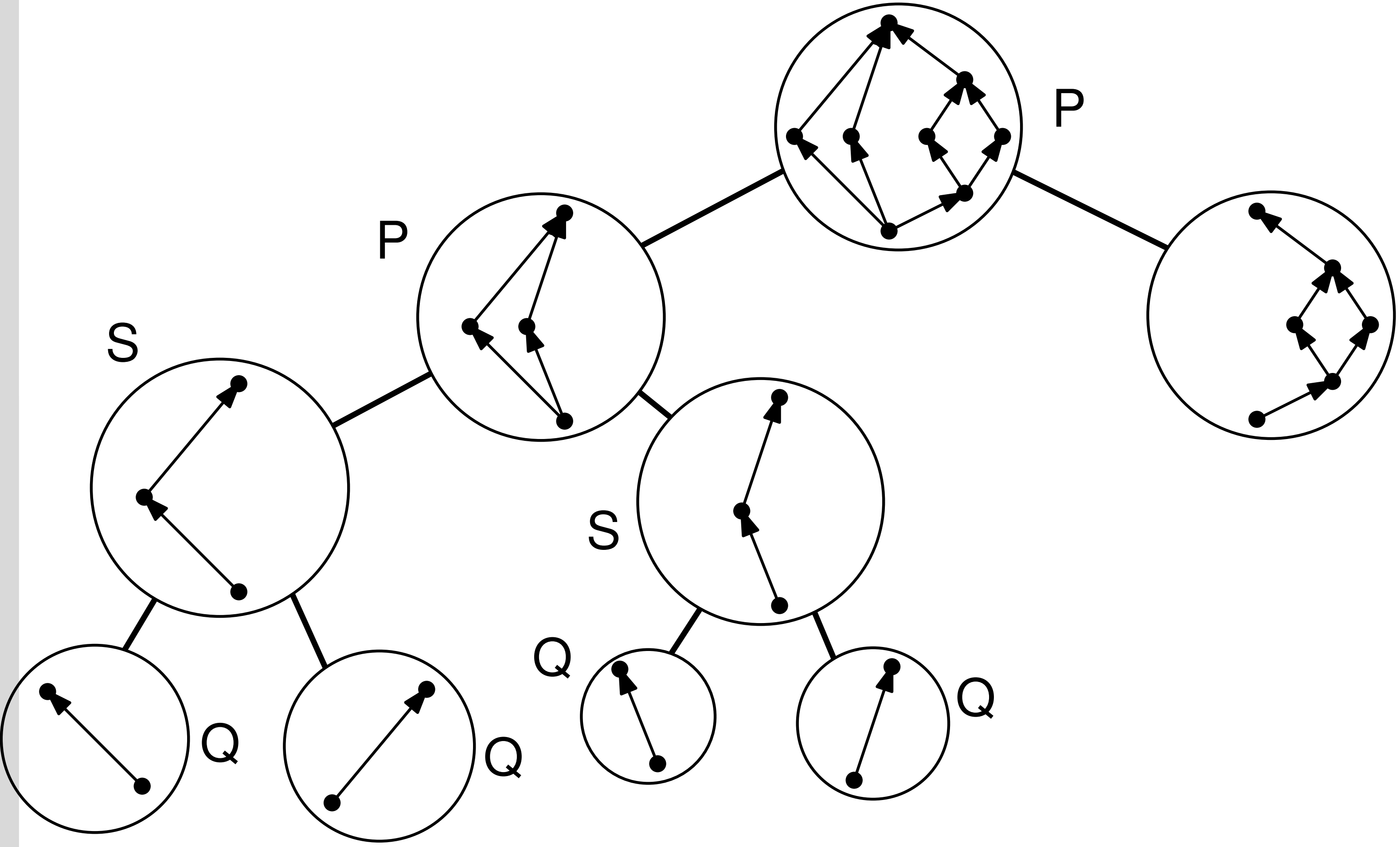


Series-parallel Graphs. Decomposition Example.



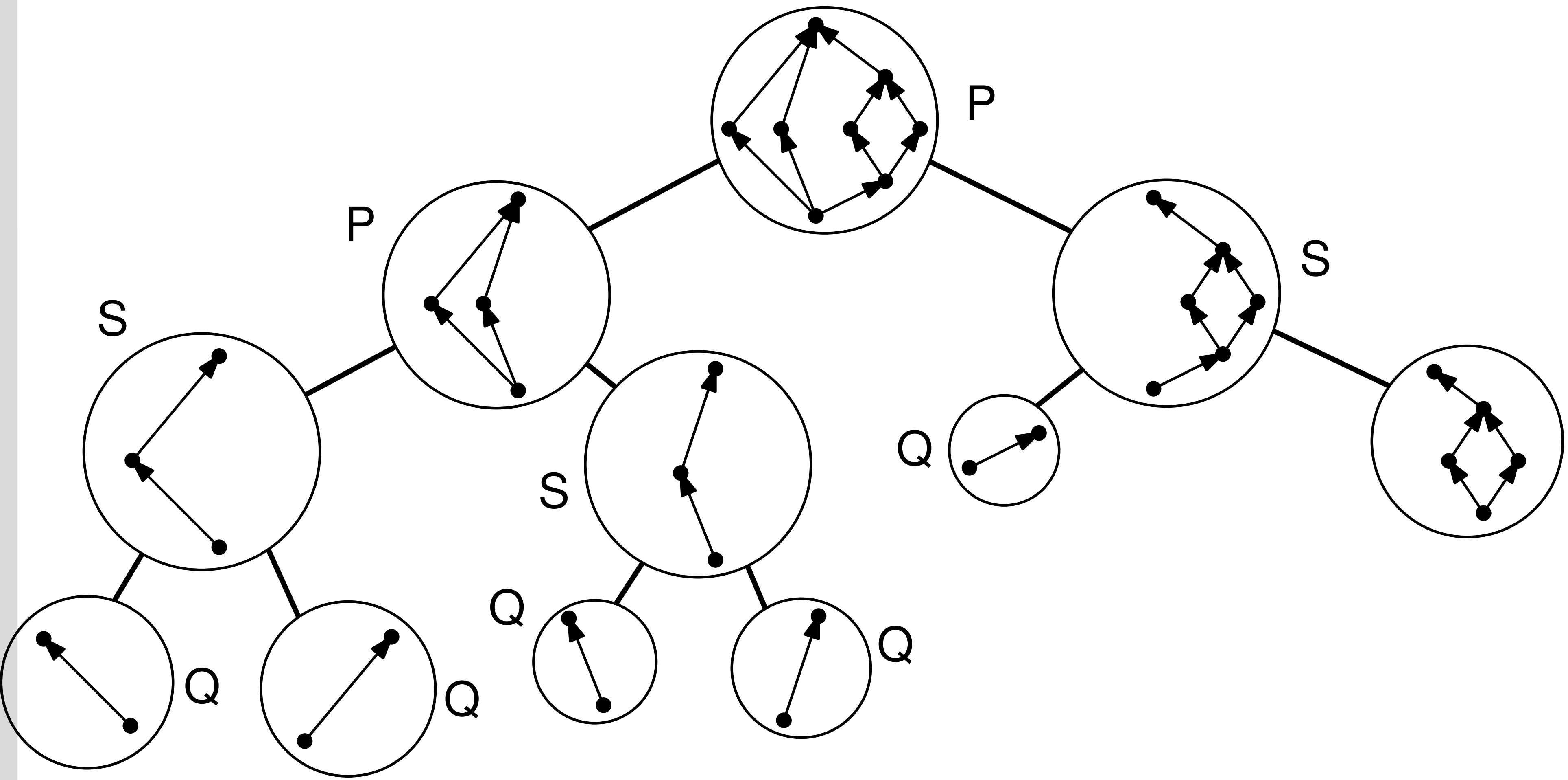
6 - 5

Series-parallel Graphs. Decomposition Example.

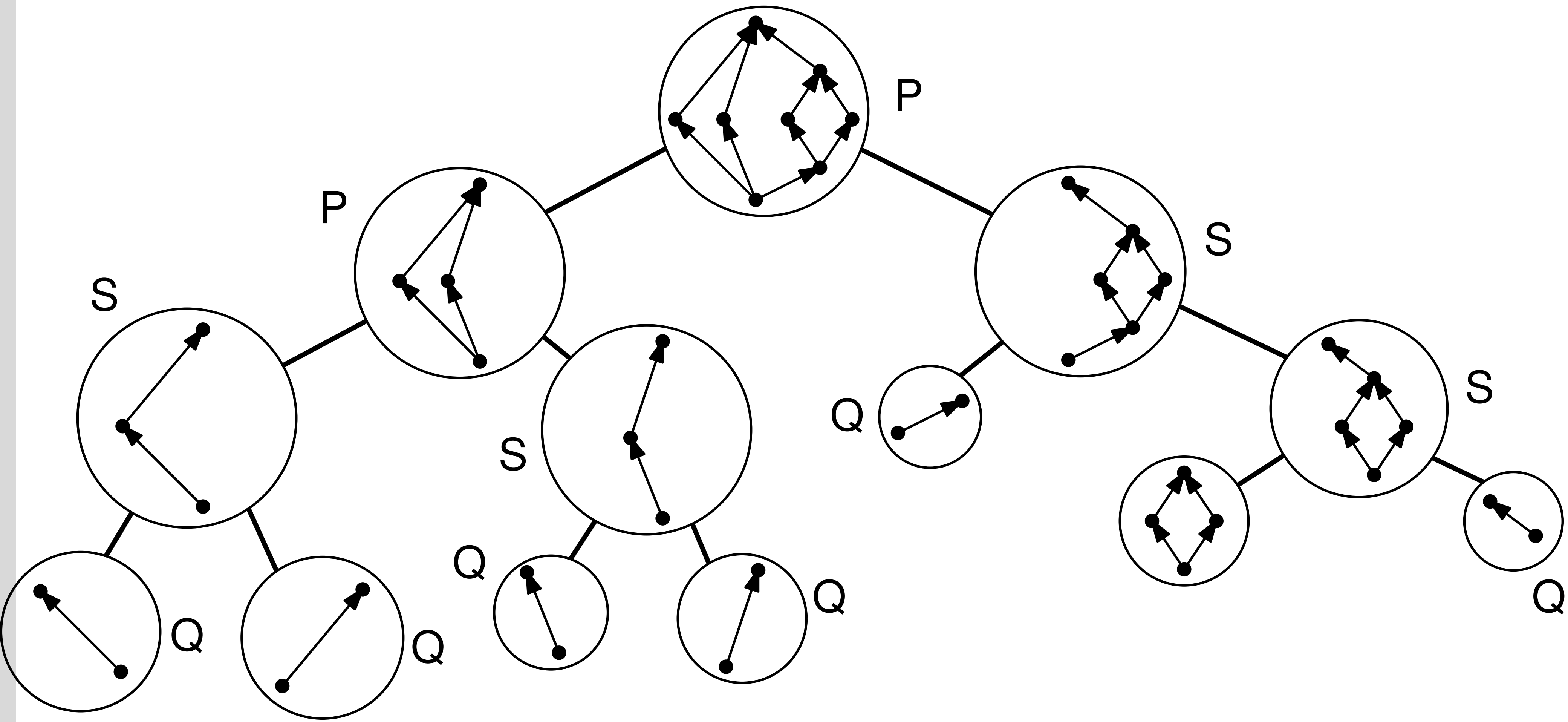


6 - 6

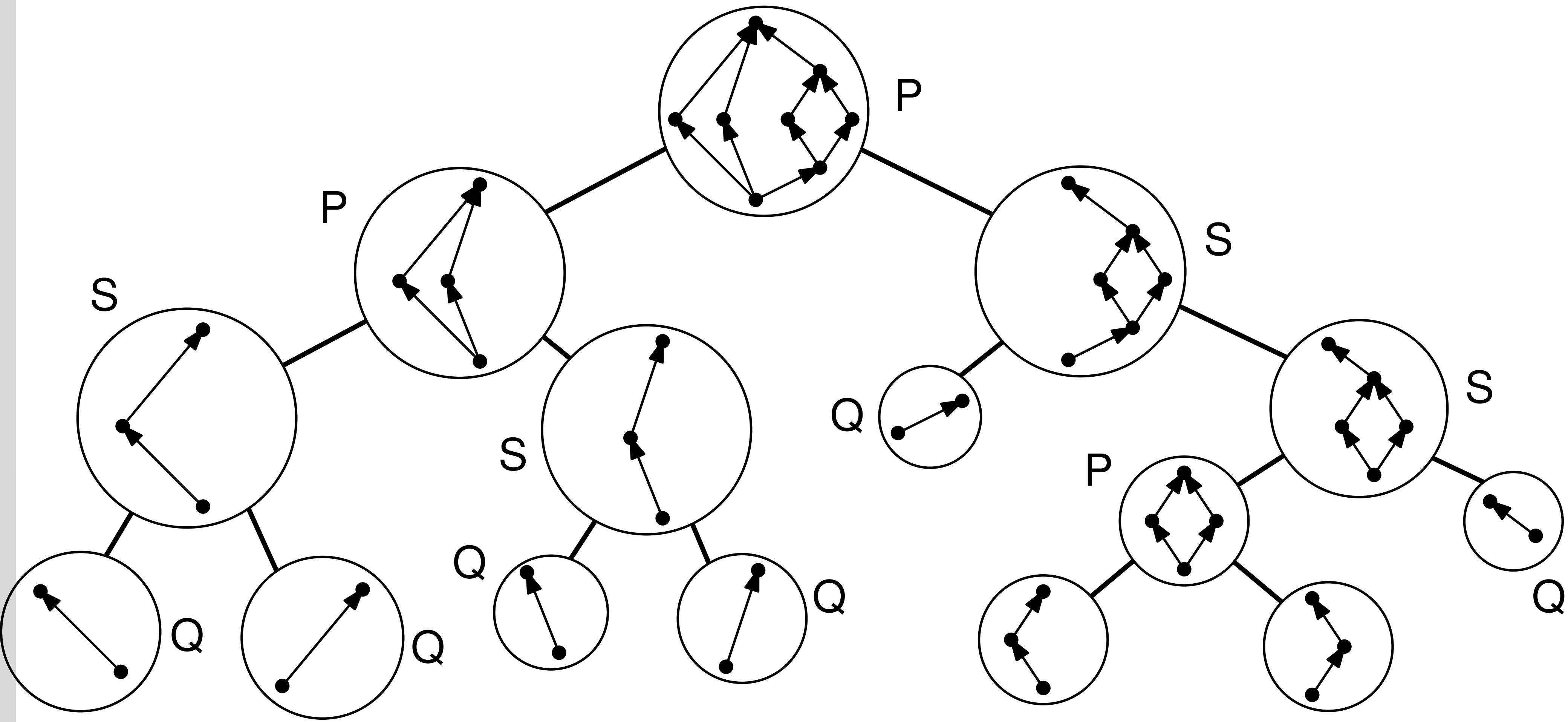
Series-parallel Graphs. Decomposition Example



Series-parallel Graphs. Decomposition Example.

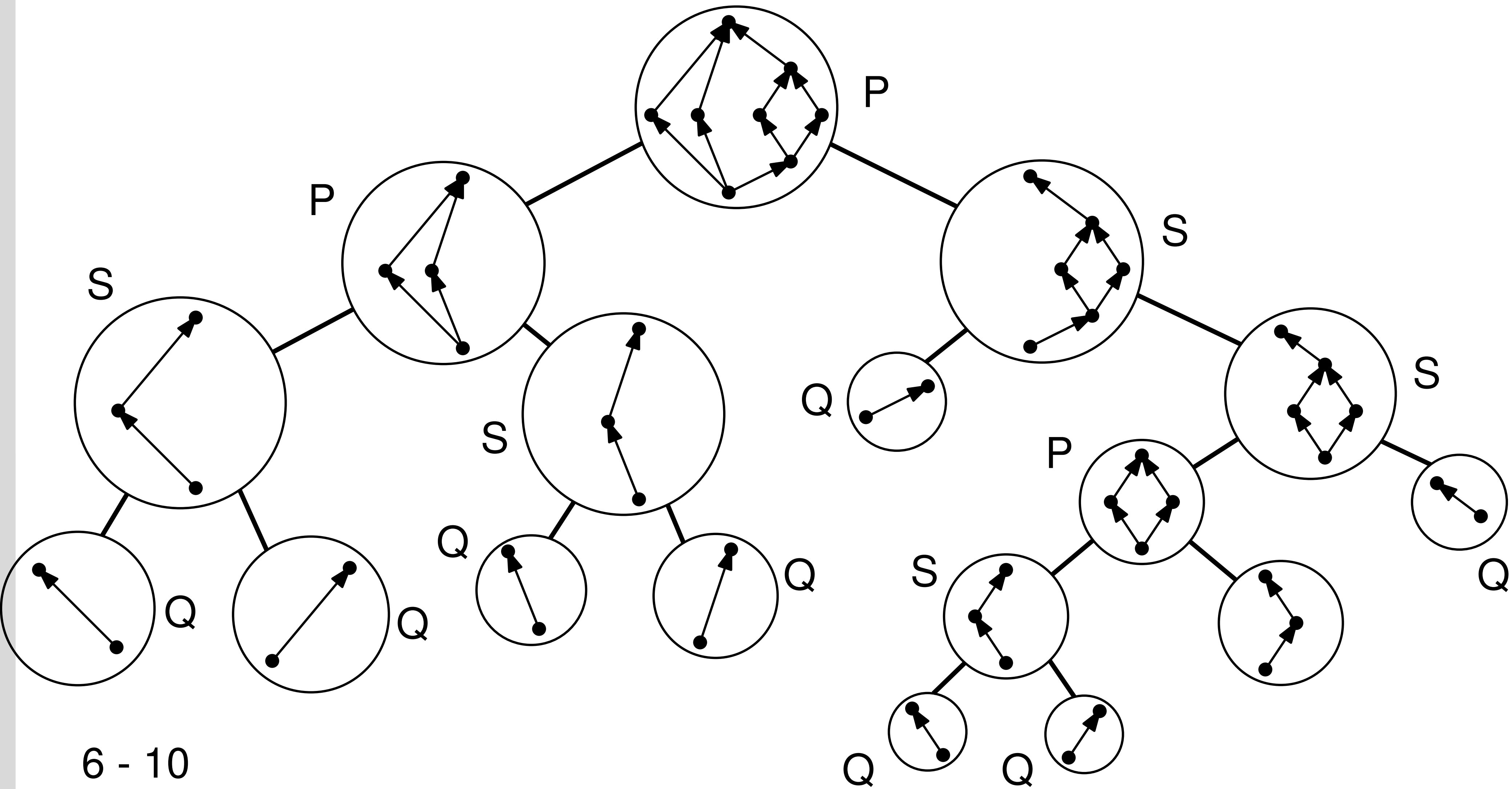


Series-parallel Graphs. Decomposition Example

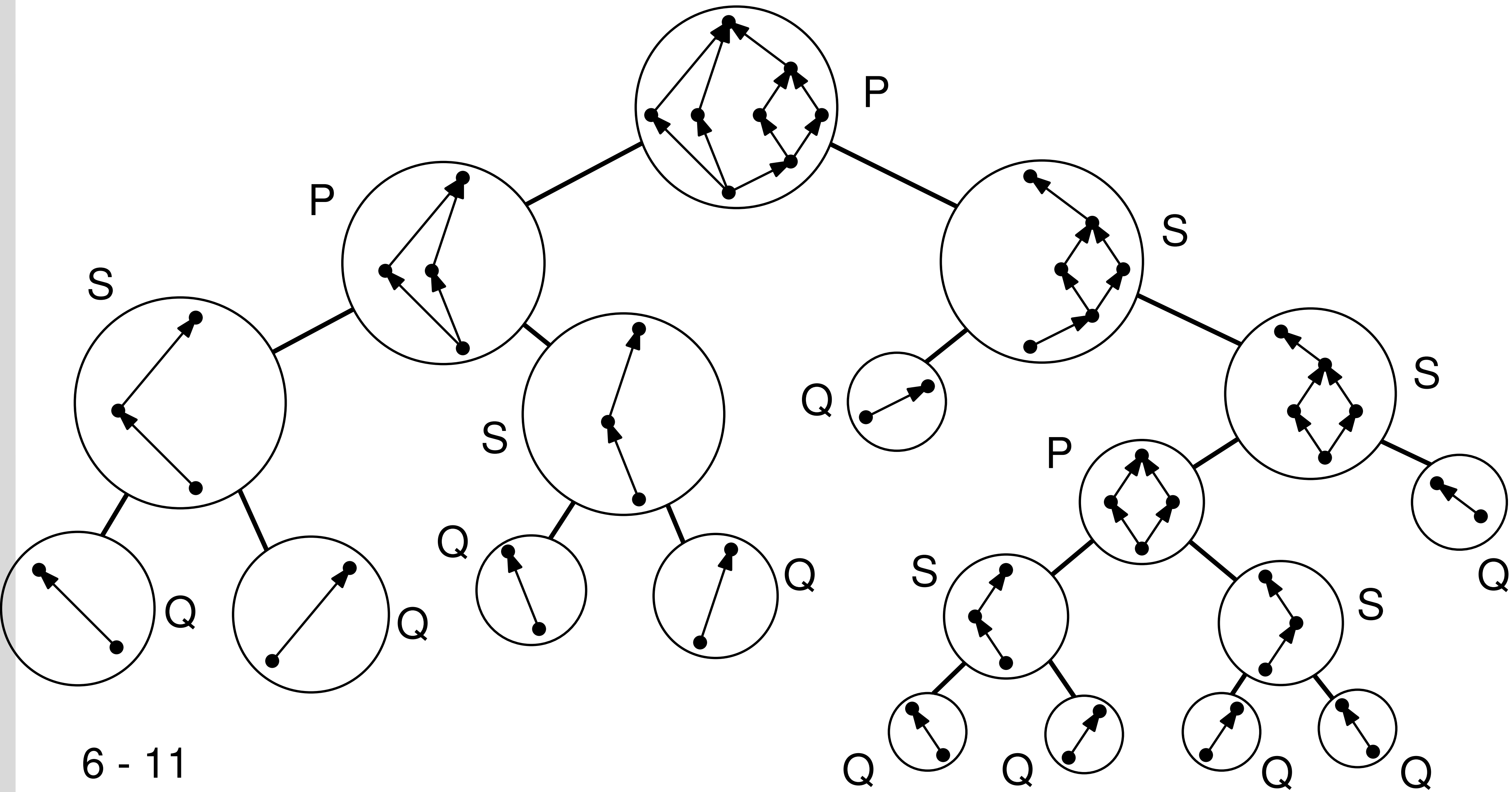


6 - 9

Series-parallel Graphs. Decomposition Example.



Series-parallel Graphs. Decomposition Example



Series-parallel Graphs. Decomposition Tree.

Karlsruhe Institute of Technology

Lemma

Series-parallel graphs are acyclic and planar.

Lemma

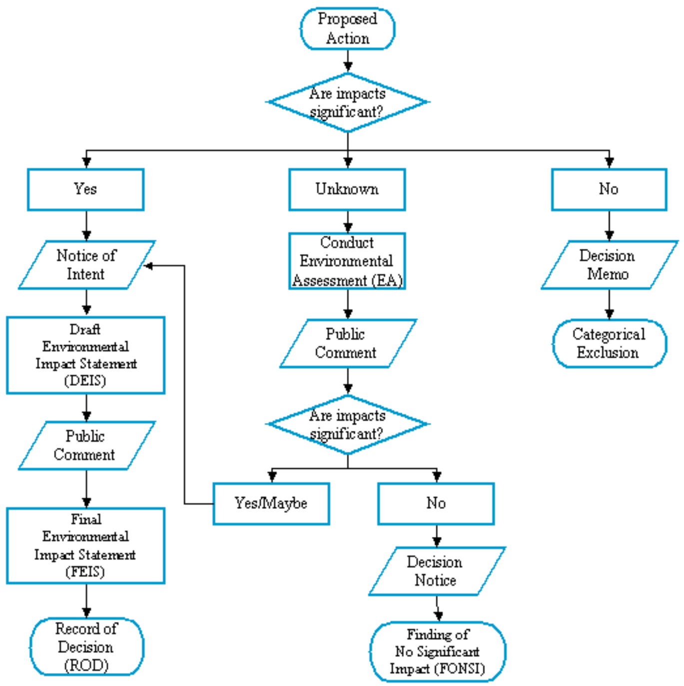
Series-parallel graphs are acyclic and planar.

Work with your neighbour(s) and then share

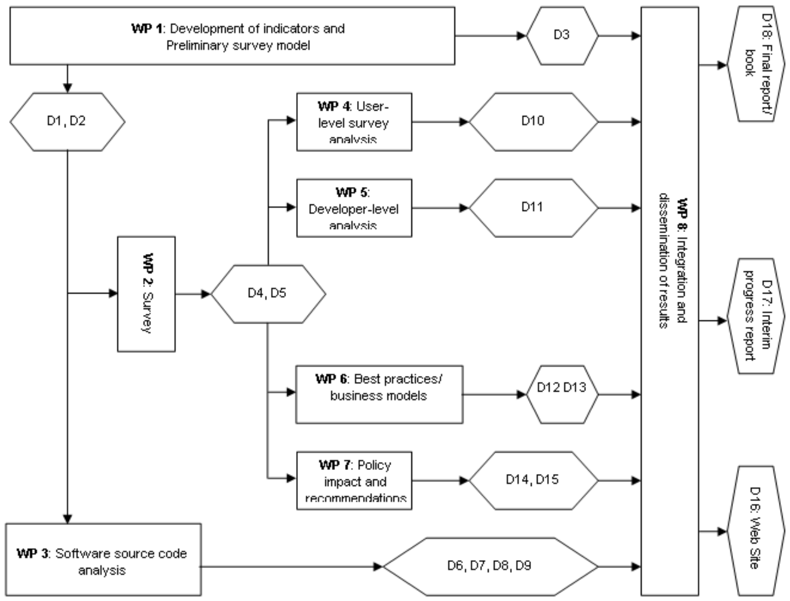
5 min

- Using the induction on the decomposition prove that a series-parallel graph is planar

Series-parallel Graphs. Applications.



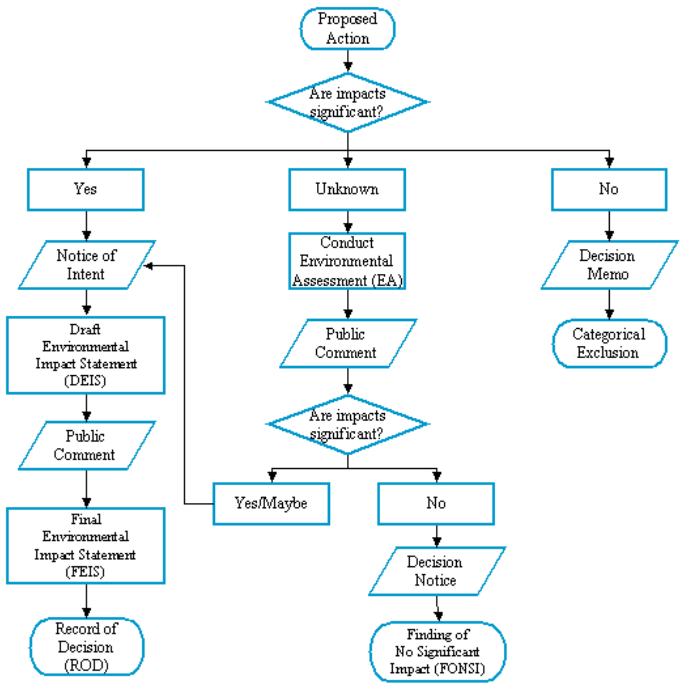
Flowcharts



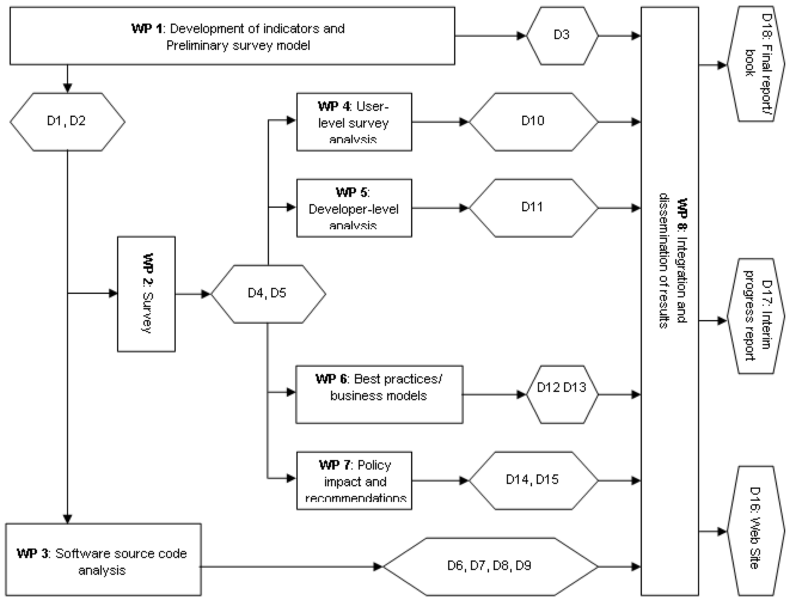
PERT-Diagrams

(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)

Overview

Series - parallel graph. Definition and Decomposition.

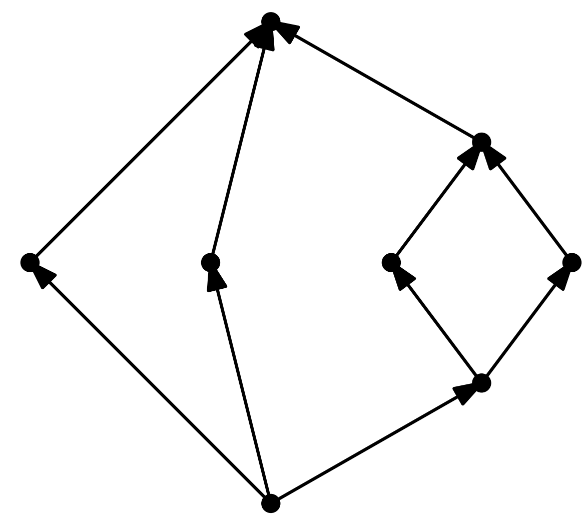
Algorithm for upward straight-line drawing.

Lower bound on the area.

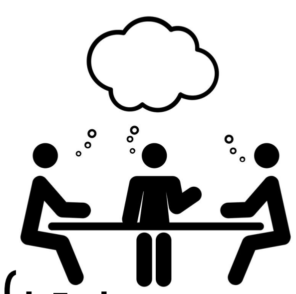
Symmetry display for series-parallel graphs.

Conventions & Aesthetics for SP-graphs

Drawing Conventions



Drawing Aesthetics

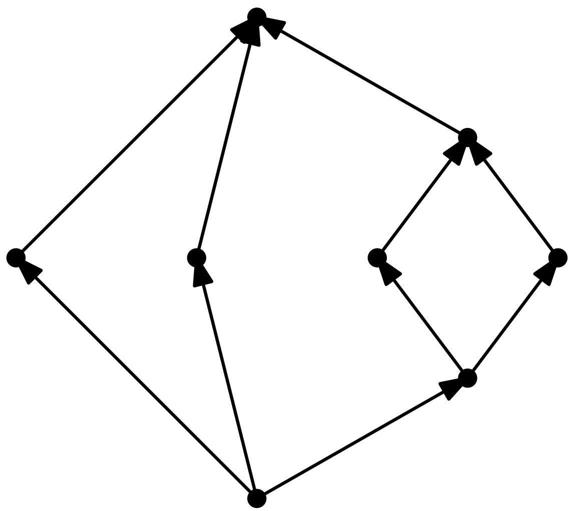


Let’s brainstorm!

Conventions & Aesthetics for SP-graphs

Drawing Conventions

- Planarity
- Straight-line edges
- Upward



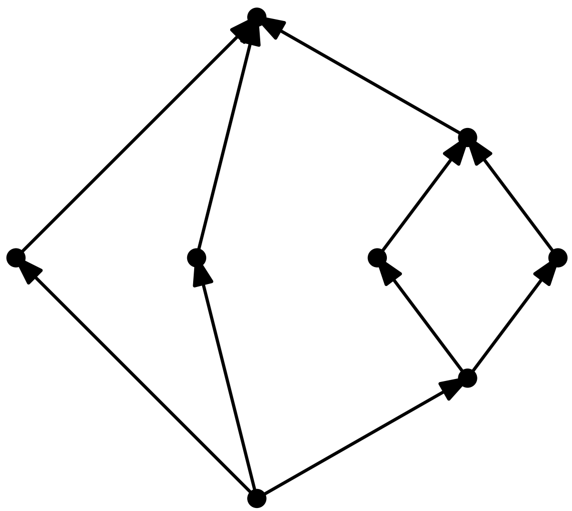
Drawing Aesthetics

Let’s brainstorm!

Conventions & Aesthetics for SP-graphs

Drawing Conventions

- Planarity
- Straight-line edges
- Upward



Drawing Aesthetics

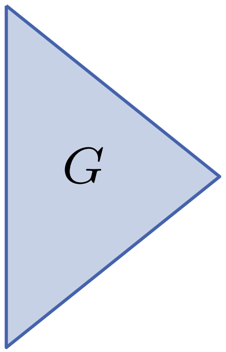
- Area
- Symmetry

Let's brainstorm!

Straight-line Drawing of SP-Graphs

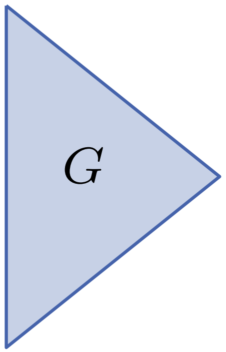
Divide & Conquer Algorithm, using the decomposition tree

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

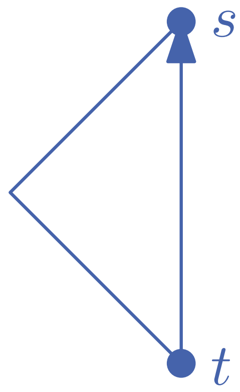


Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree

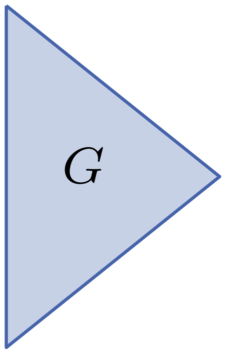


- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):

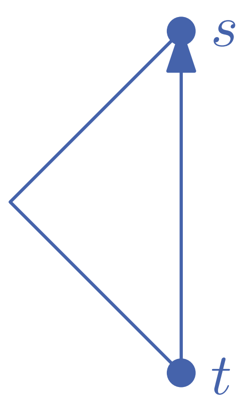


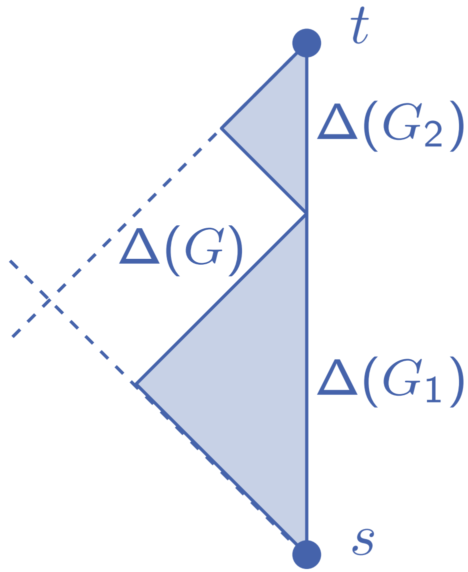
Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree



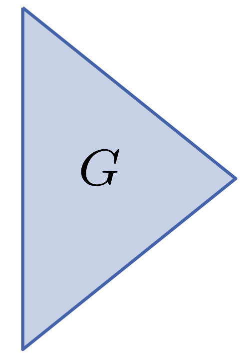
- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)



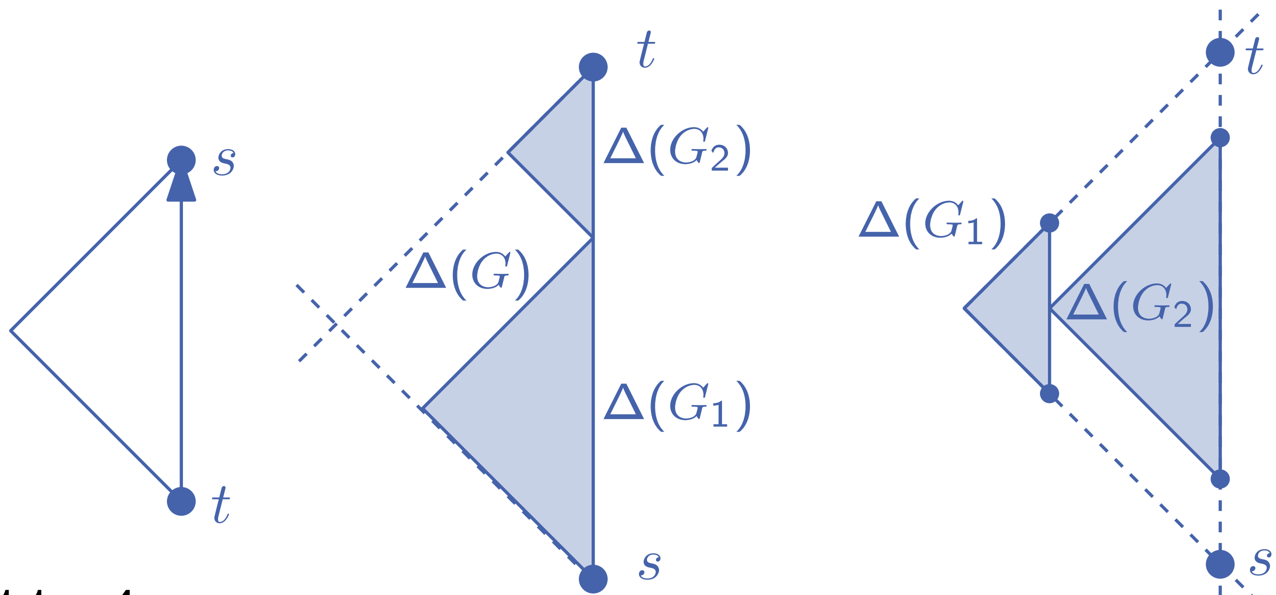


Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree

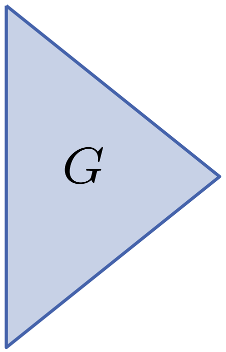


- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)

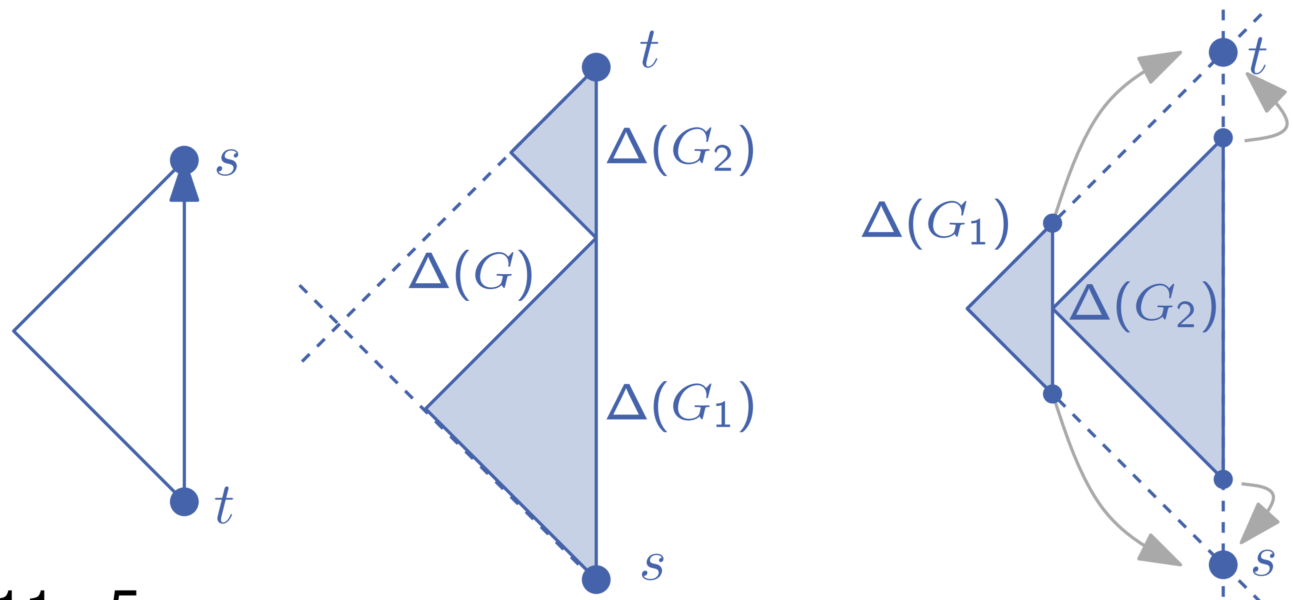


Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree



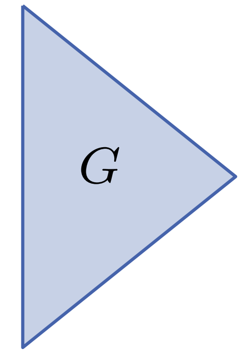
- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)



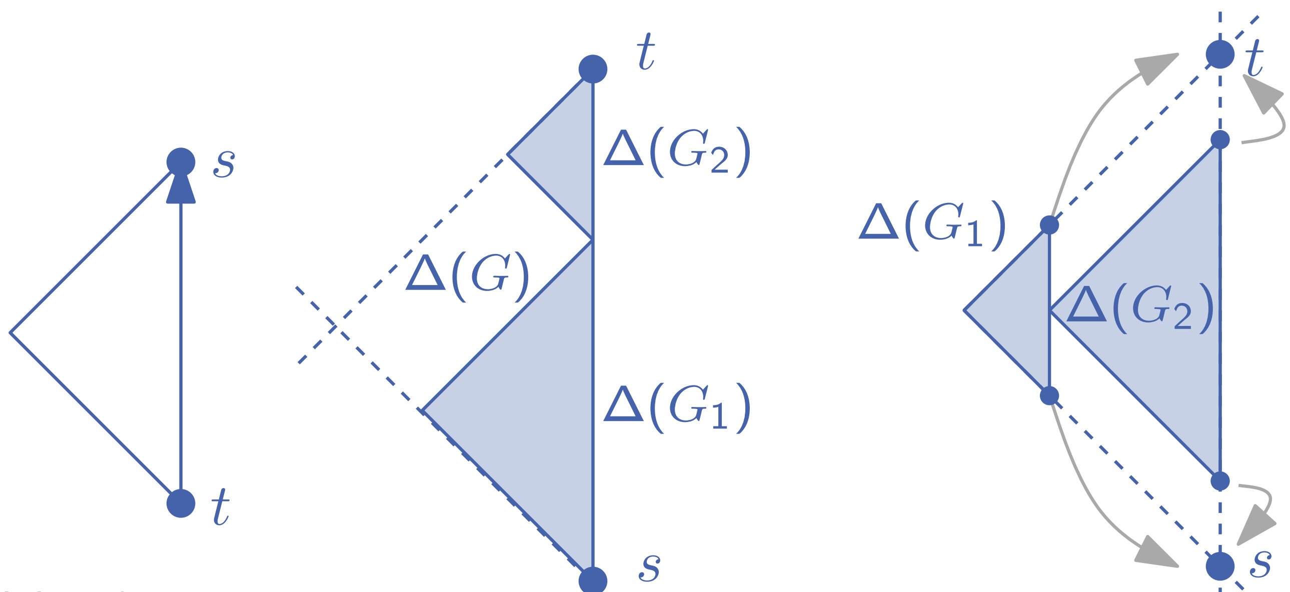
11 - 5

Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree



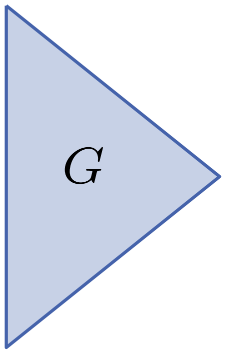
- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)



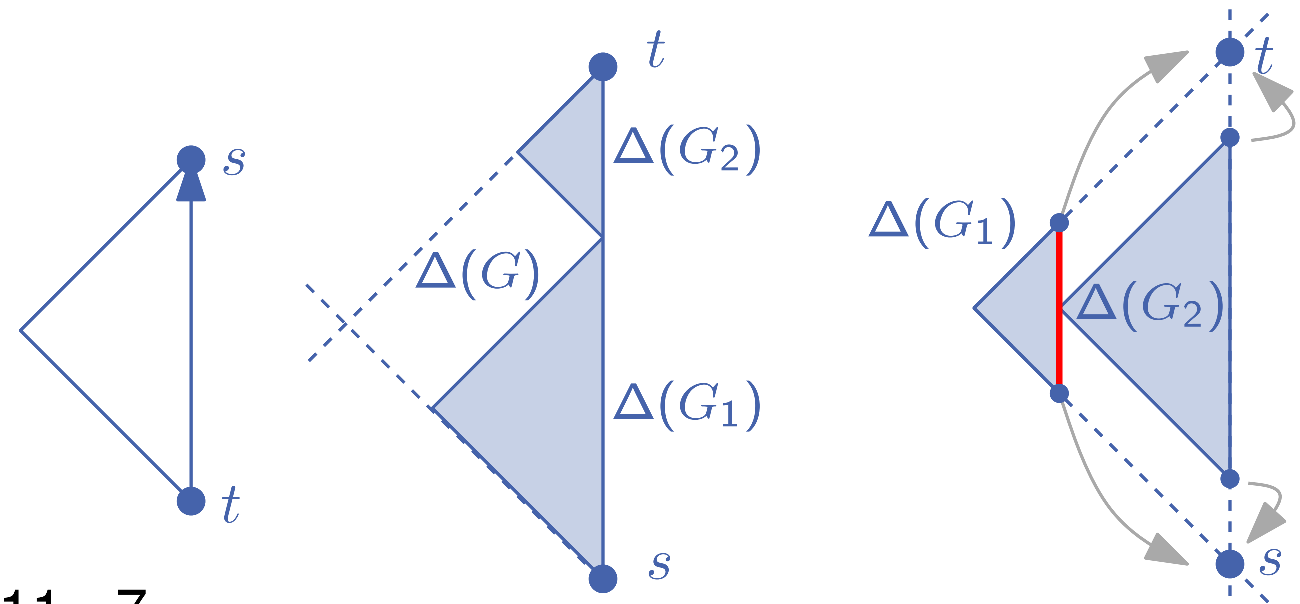
Do you see any problem?

Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree

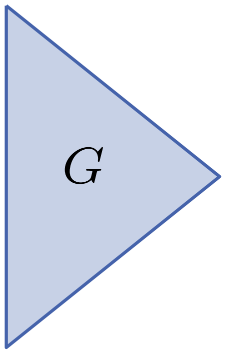


- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)

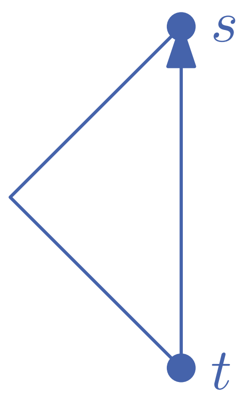


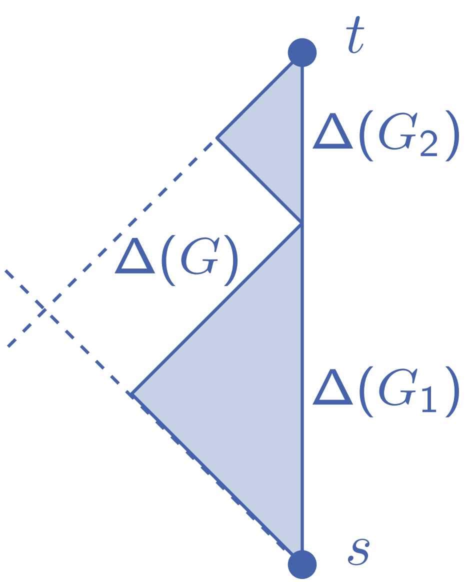
Straight-line Drawing of SP-Graphs

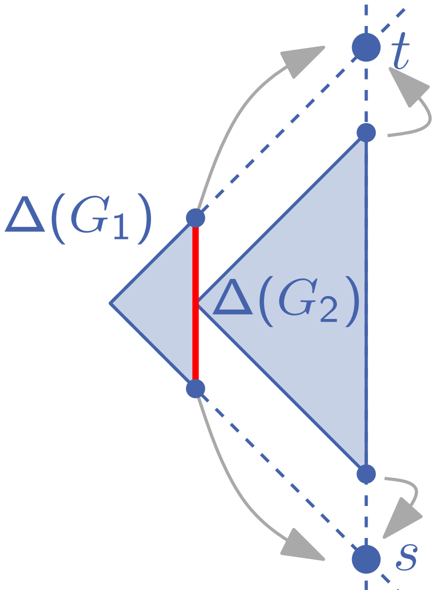
Divide & Conquer Algorithm, using the decomposition tree



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



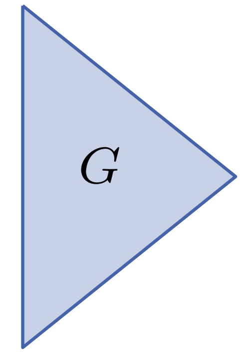




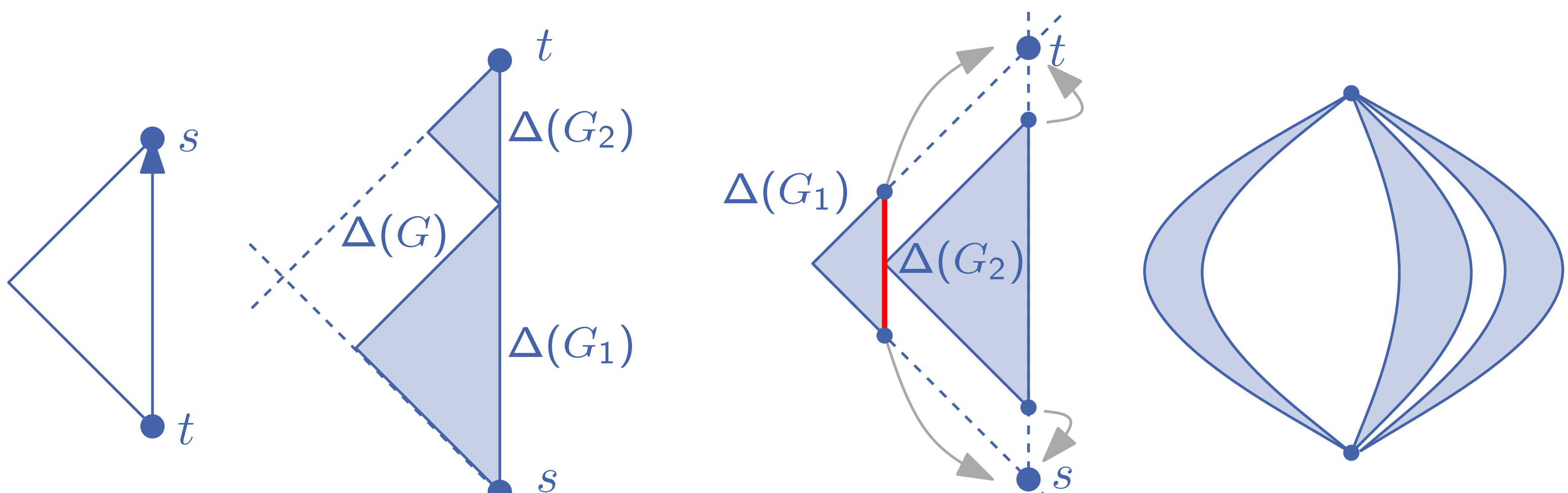
change embedding!

Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree



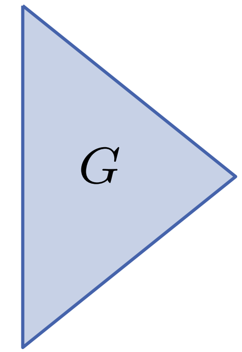
- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)



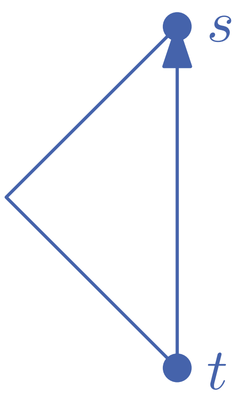
change embedding!

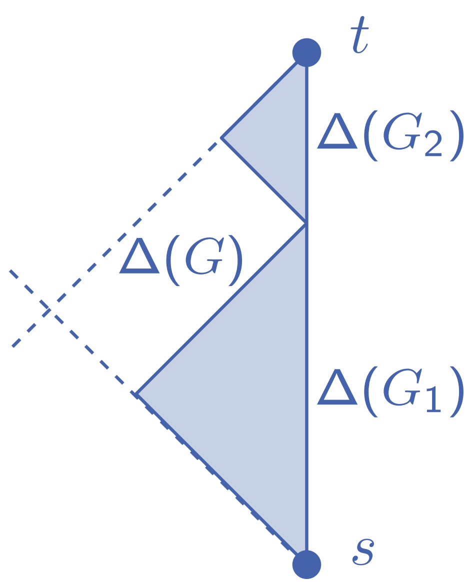
Straight-line Drawing of SP-Graphs

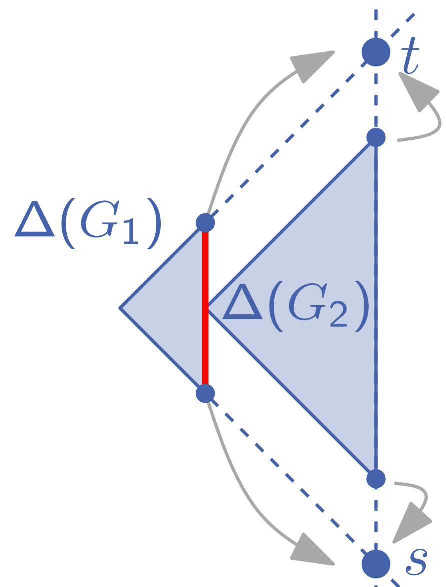
Divide & Conquer Algorithm, using the decomposition tree

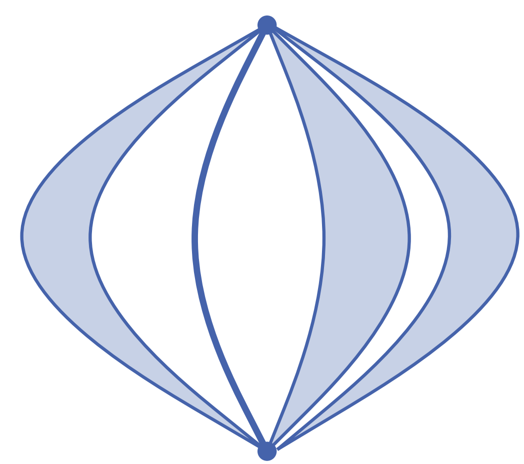


- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)



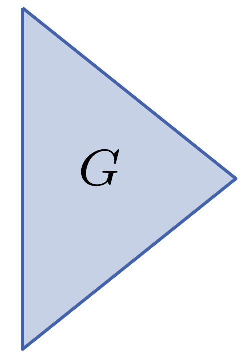




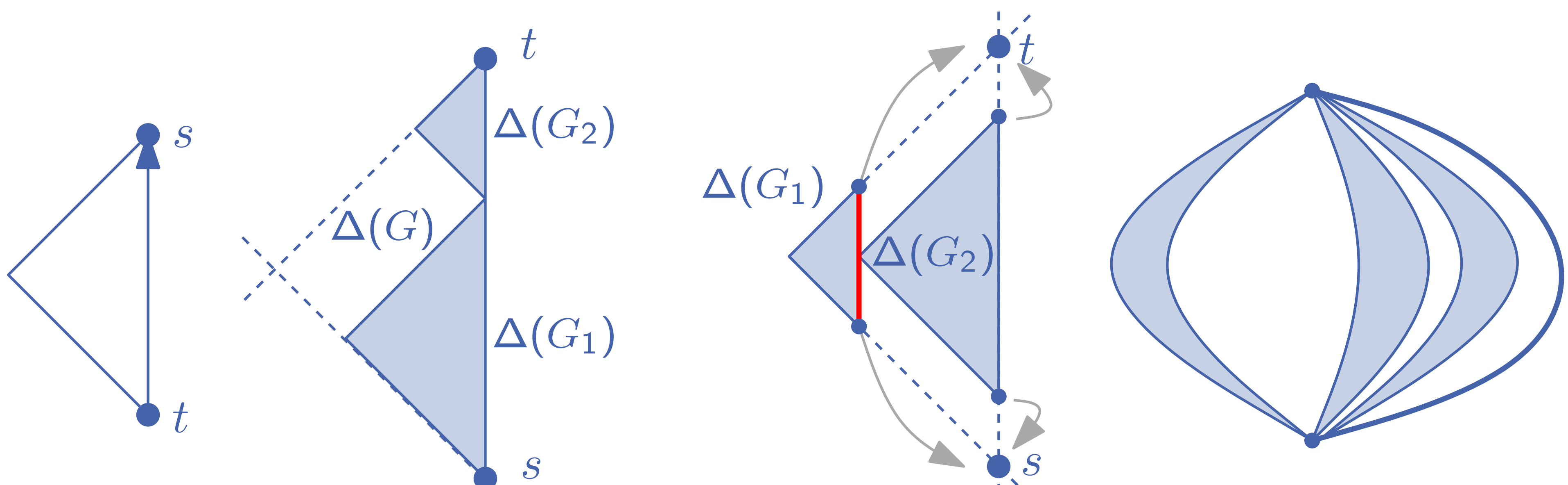


change embedding!

Divide & Conquer Algorithm, using the decomposition tree



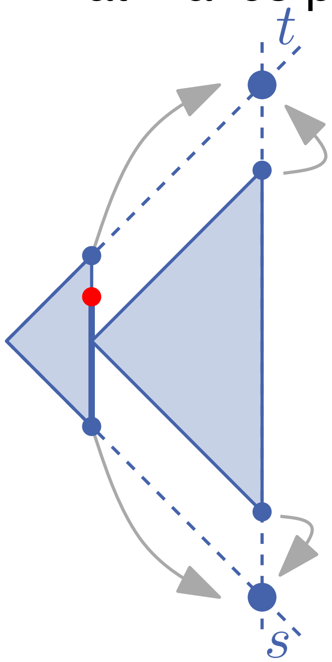
- Draw graph G inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)



change embedding!

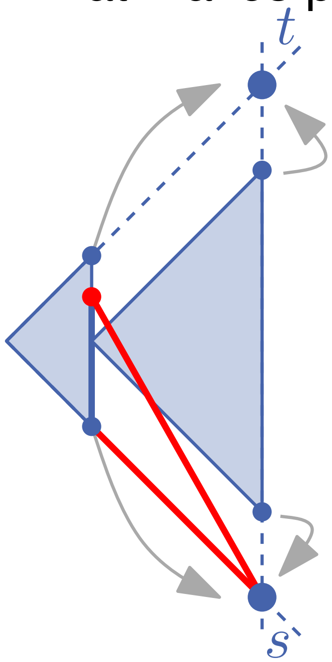
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



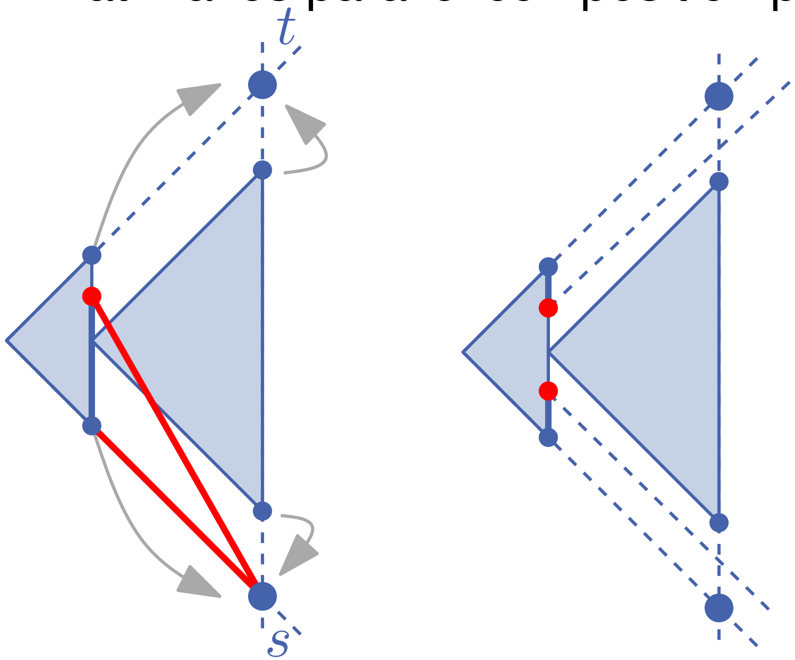
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



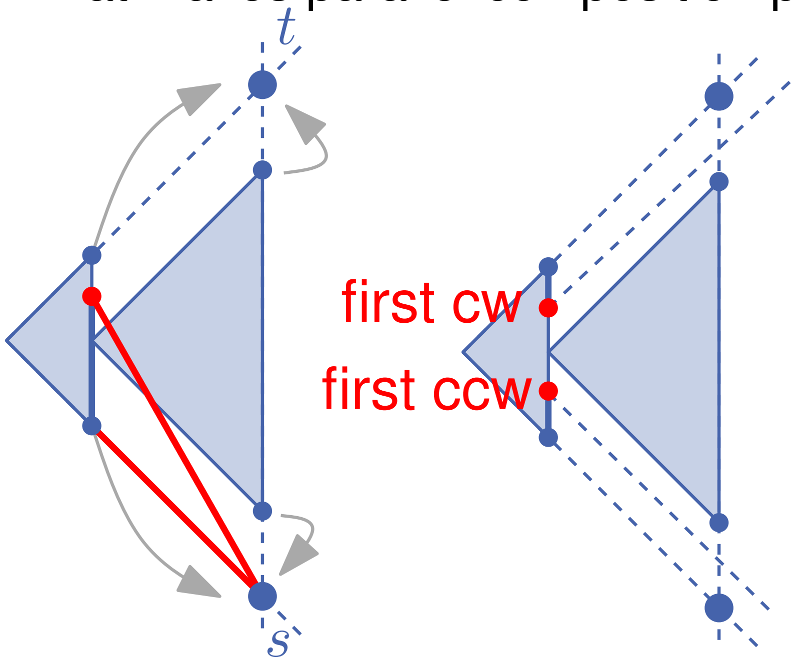
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



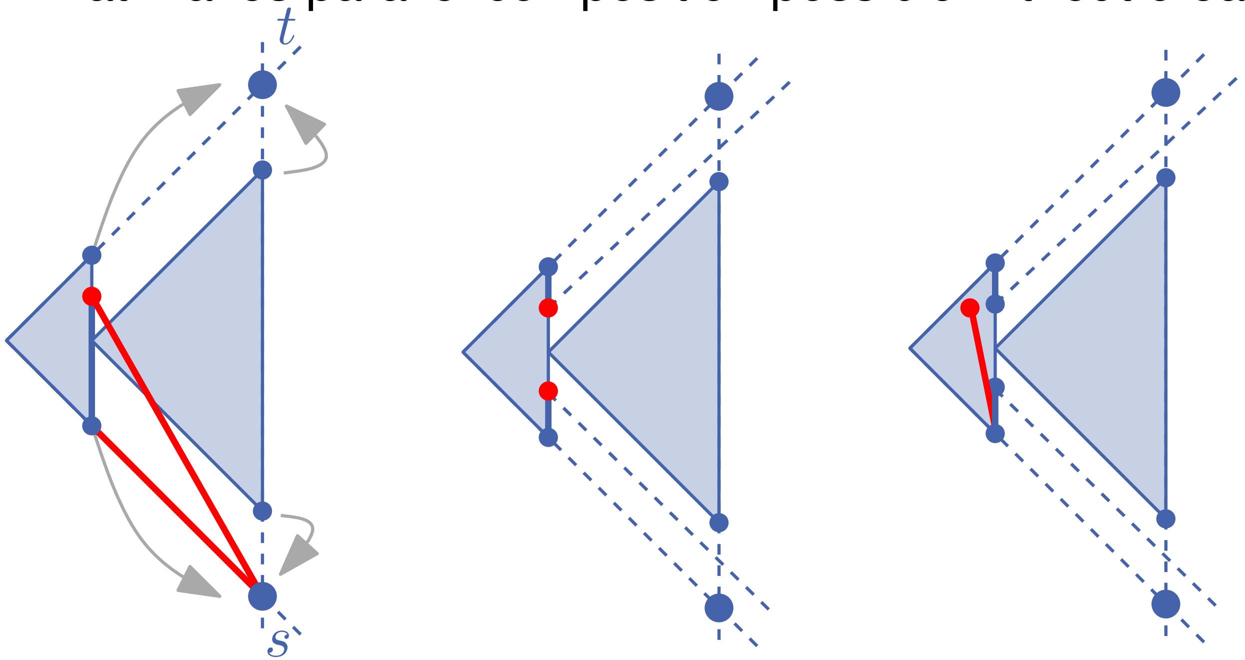
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



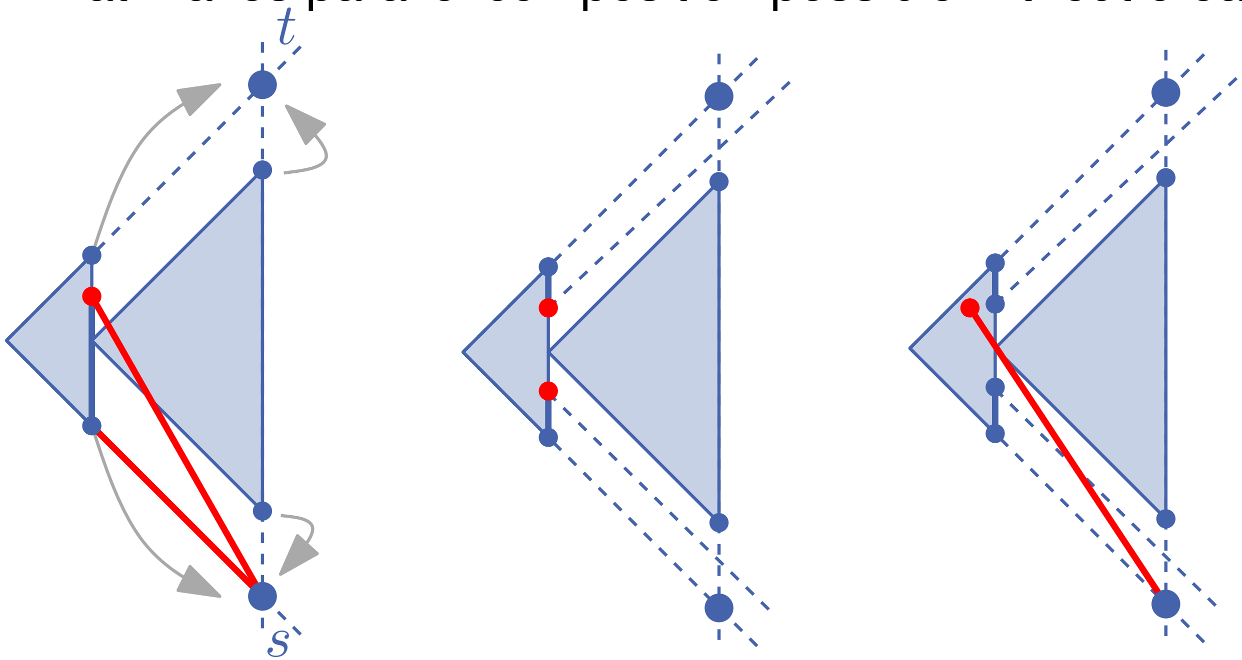
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



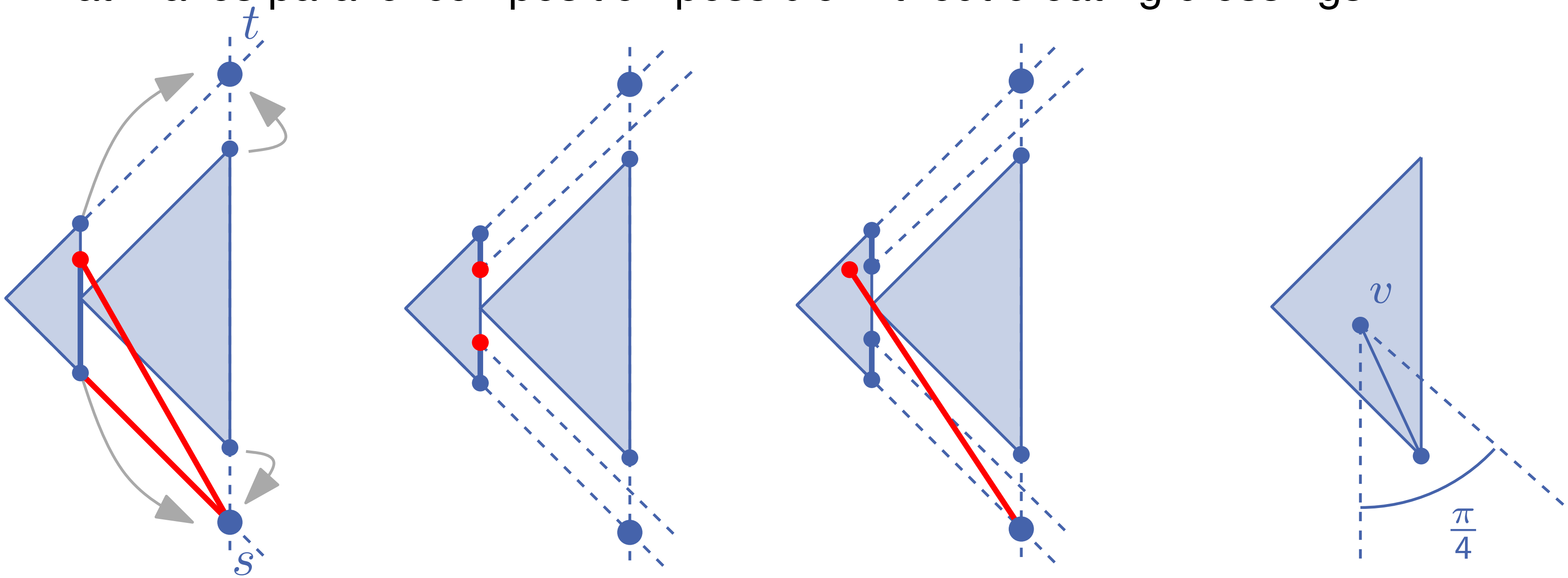
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



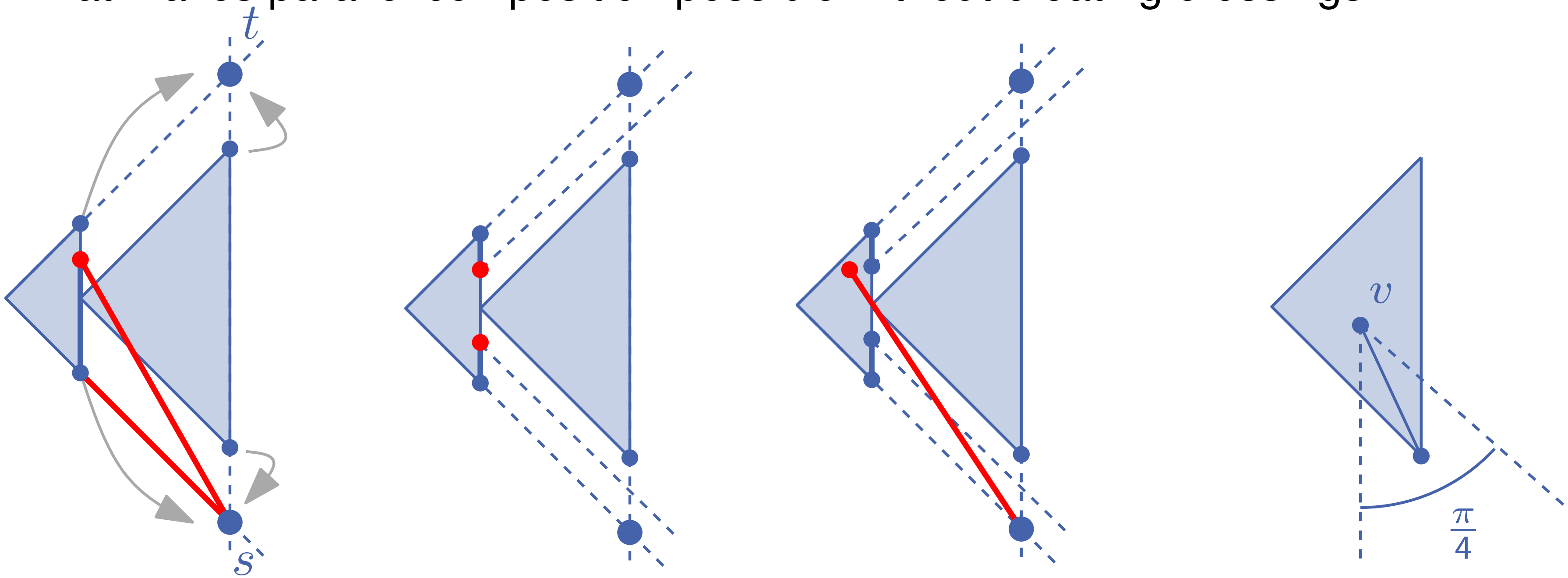
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



Straight-line Drawing of SP-Graphs

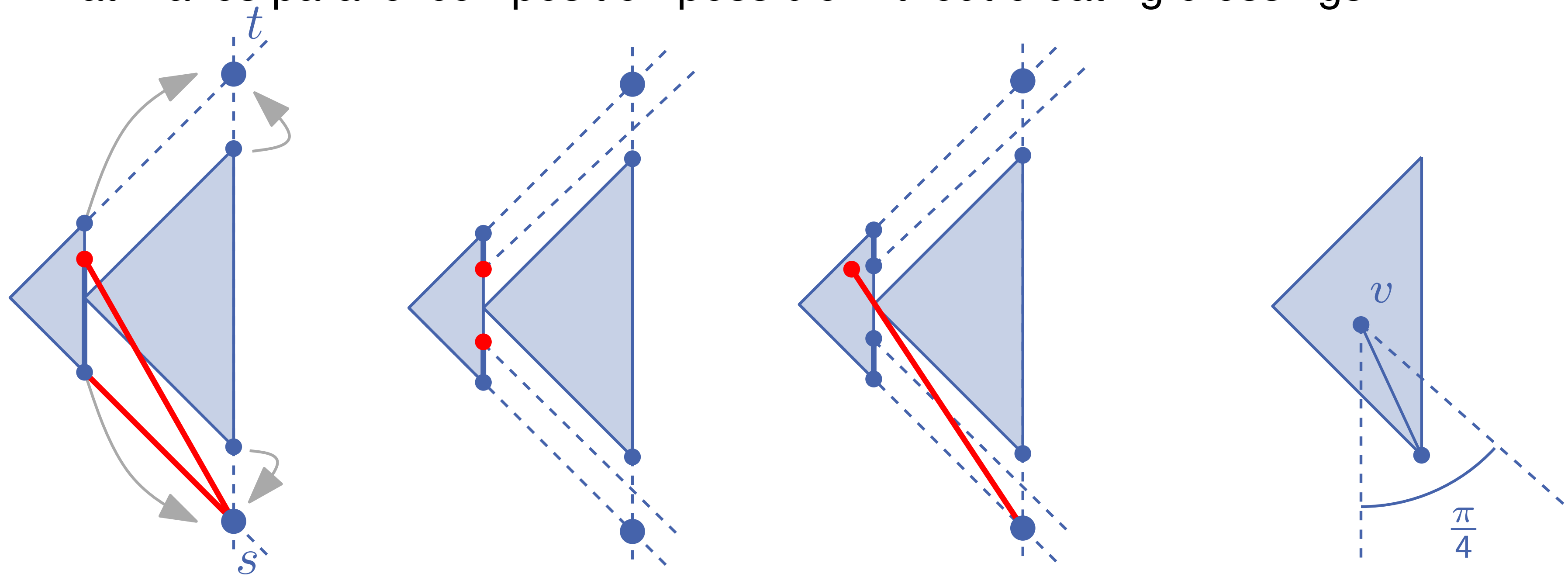
- What makes parallel composition possible without creating crossings?



Assume the following holds: angle(v) does not contain any vertex

Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?

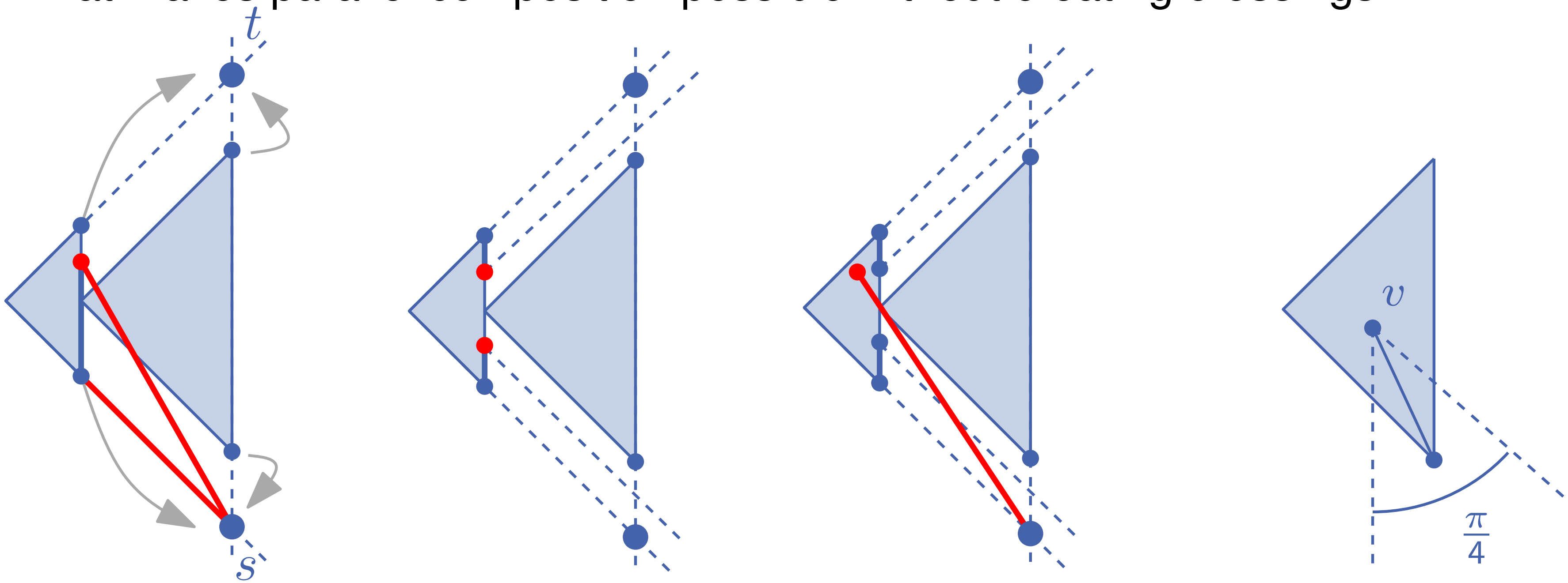


Assume the following holds: angle(v) does not contain any vertex

- This condition **is** preserved during the induction step.

Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?



Assume the following holds: angle(v) does not contain any vertex

- This condition **is** preserved during the induction step.

Lemma

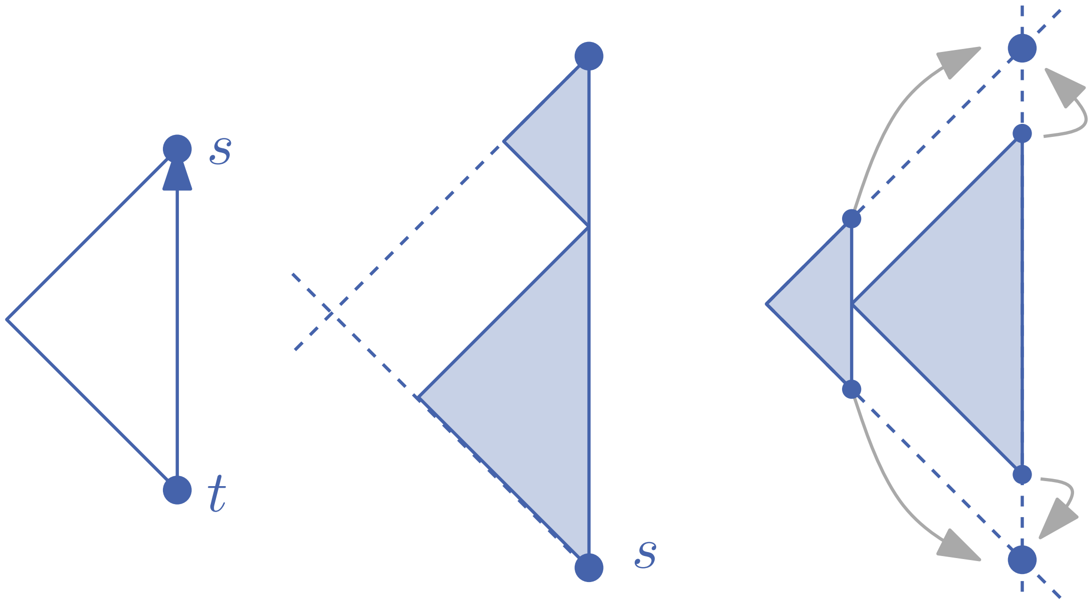
The drawing produced by the algorithm is planar.

Straight-line Drawing of SP-Graphs

Work with your neighbour(s) and then share

3 min

- What is the asymptotic upper bound on the area of the drawing? **Hint:** Think in terms of edges.



Straight-line Drawing of SP-Graphs

Work with your neighbour(s) and then share

3 min

- What is the asymptotic upper bound on the area of the drawing? **Hint:** Think in terms of edges.

Lemma

The area of the produced drawing is $O(m^2)$, m is the number of edges.

Theorem

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

Overview

Series - parallel graph. Definition and Decomposition.

Algorithm for upward straight-line drawing.

Lower bound on the area.

Symmetry display for series-parallel graphs.

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 $\Omega(4^n)$.

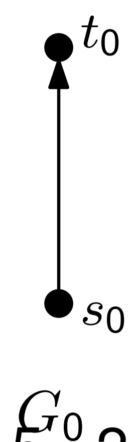
Institut für Theoretische Informatik
Lehrstuhl Algorithmik I

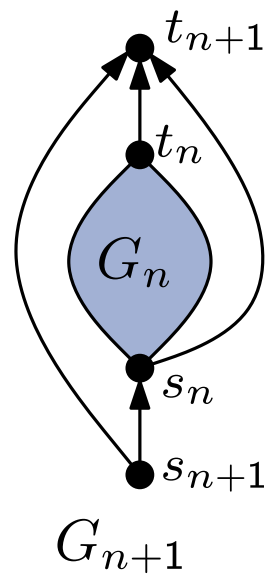
Lower Bound for the Area

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:



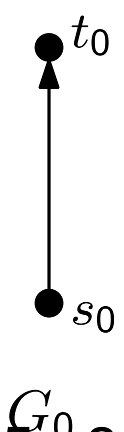


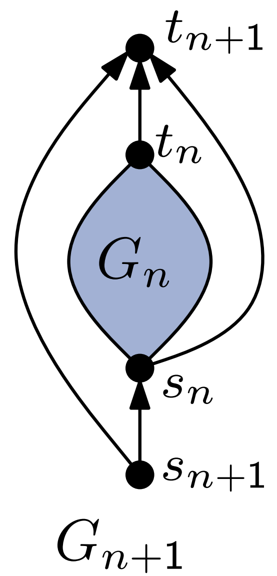
Lower Bound for the Area

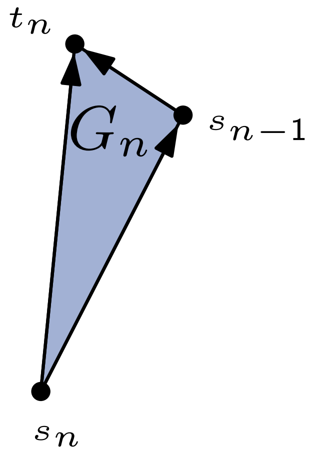
Theorem [Bertolazzi et al. 94]

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:



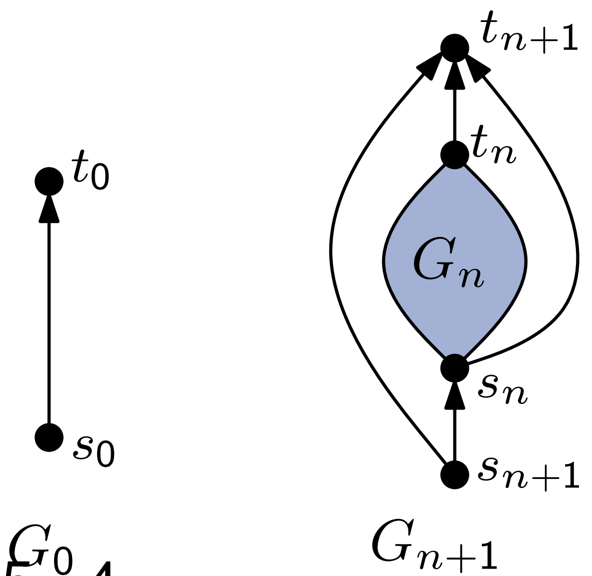


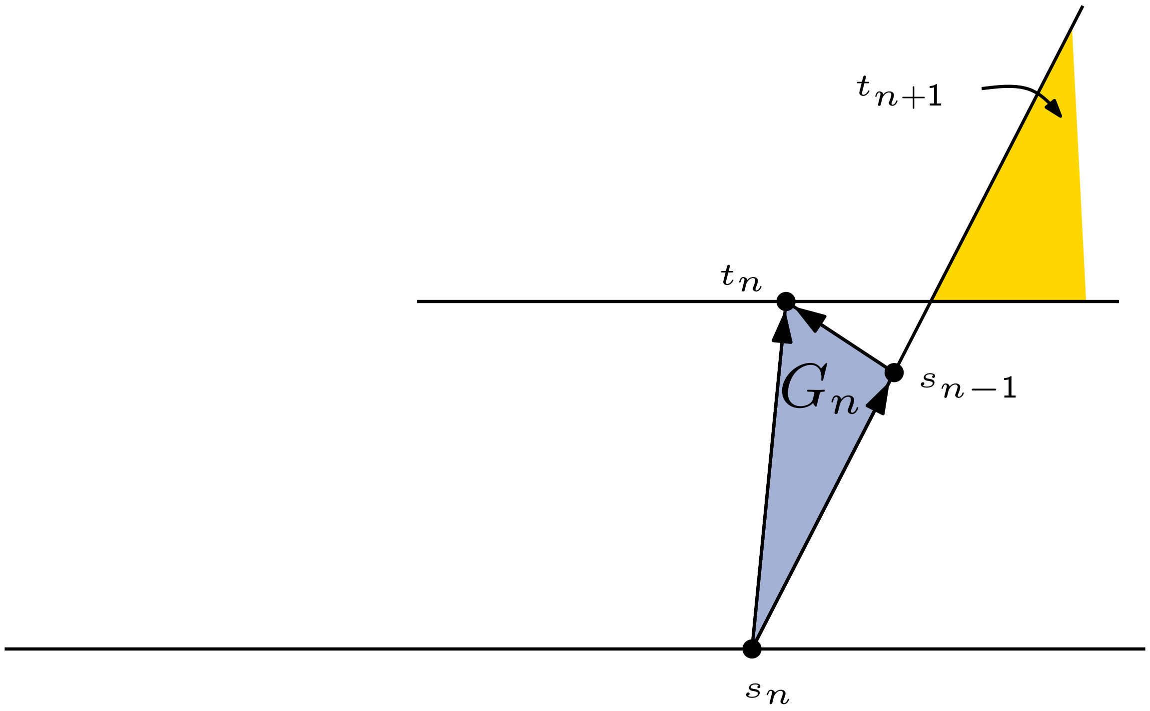


Theorem [Bertolazzi et al. 94]

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:



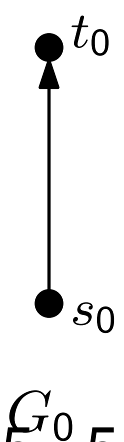


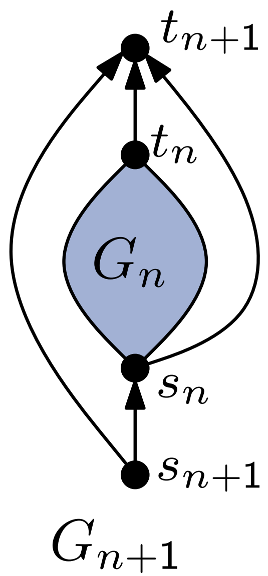
Lower Bound for the Area

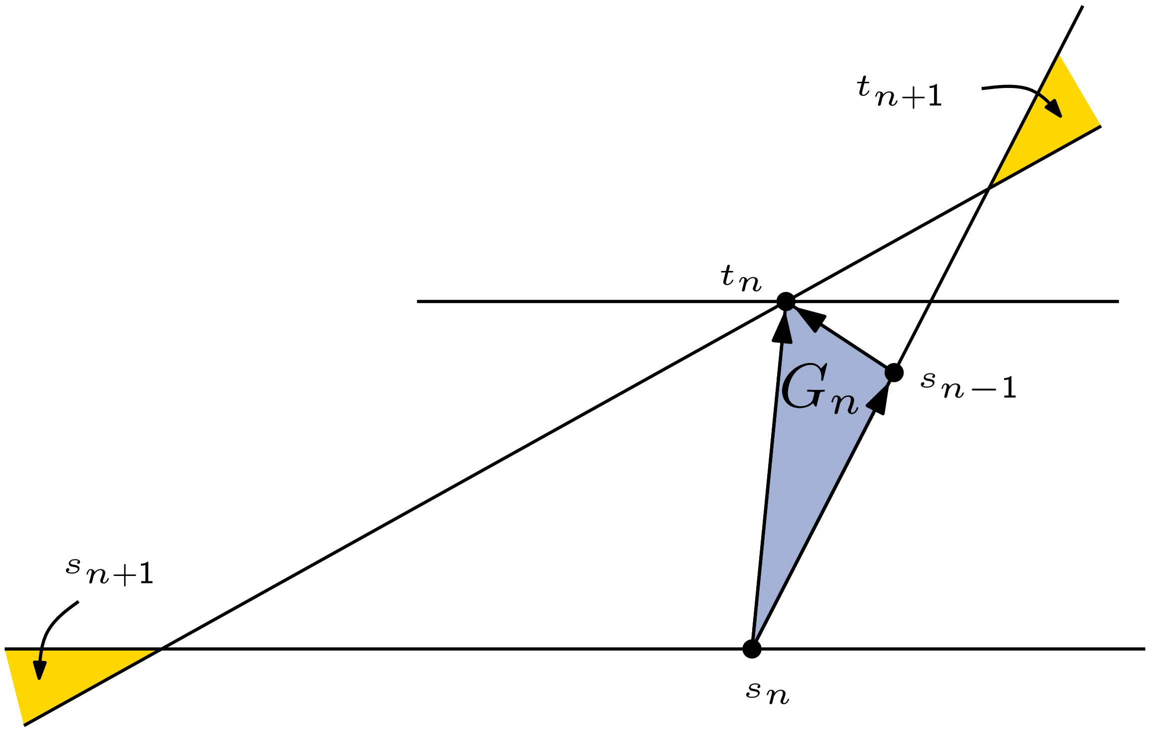
Theorem [Bertolazzi et al. 94]

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:







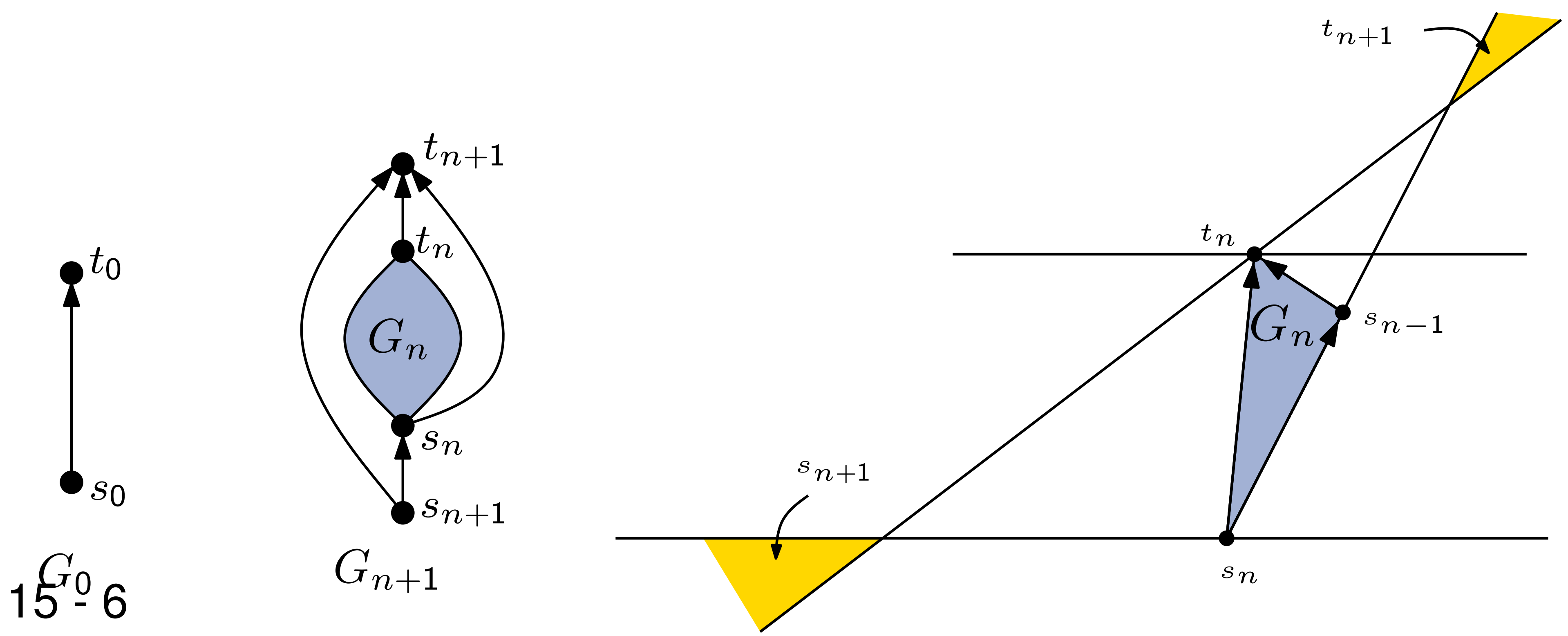
15 - 5

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:

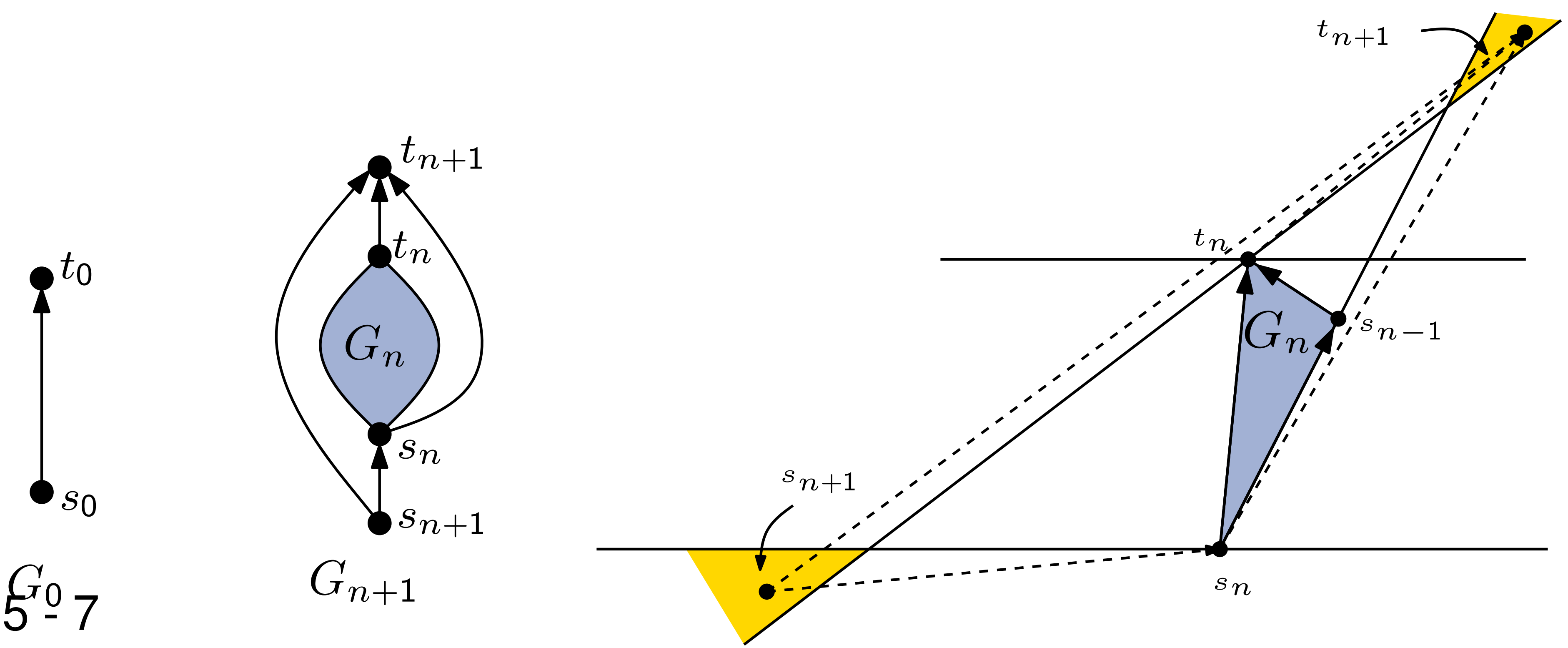


Lower Bound for the Area

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:



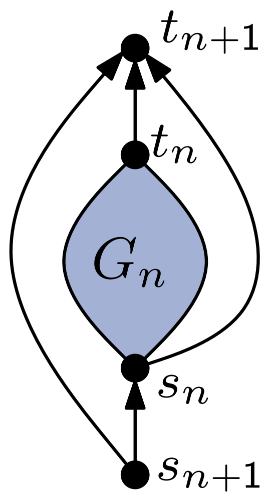
Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

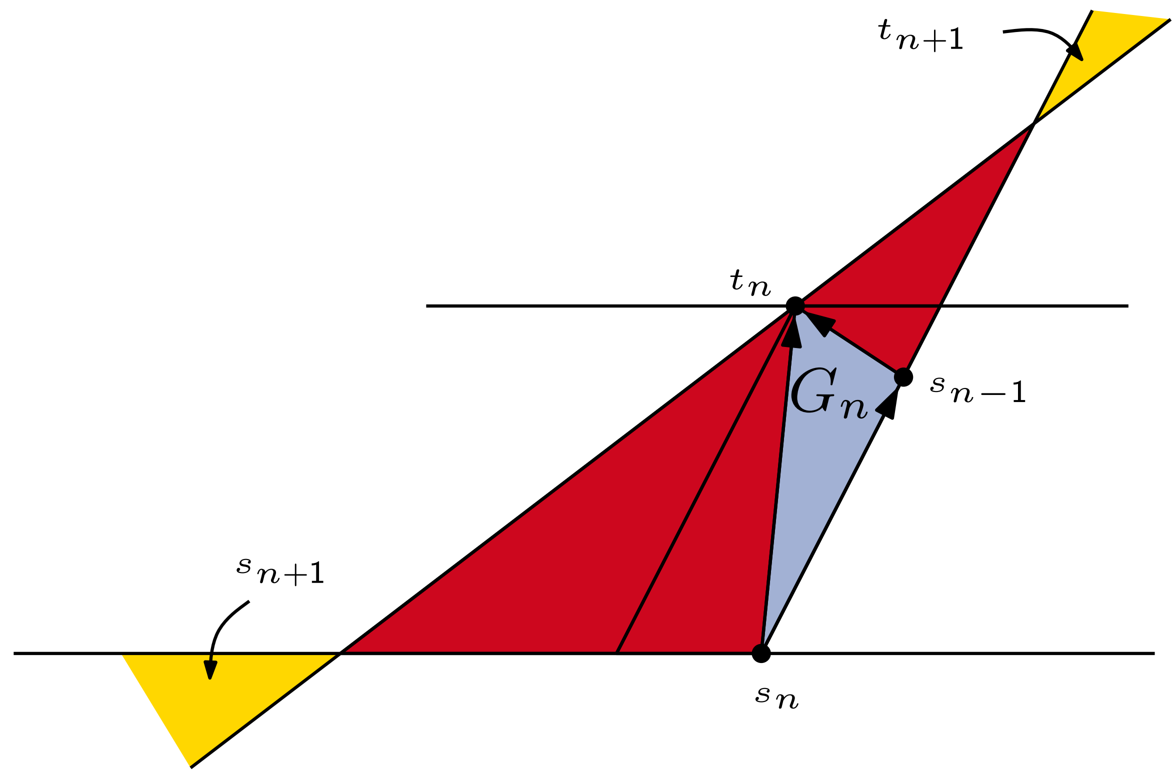
There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:

G_0



G_{n+1}

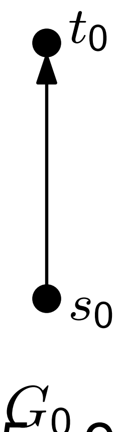


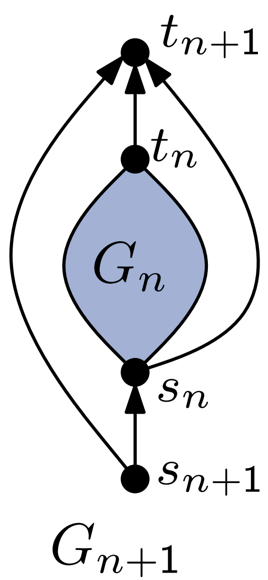
Lower Bound for the Area

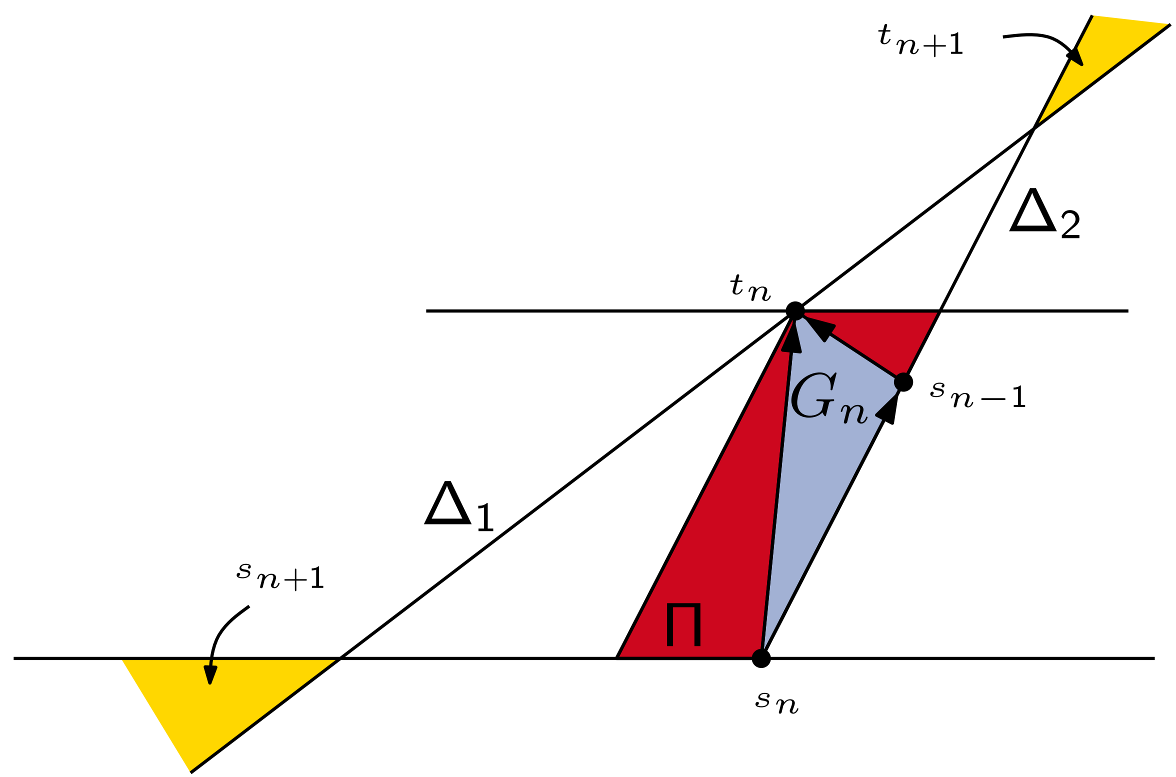
Theorem [Bertolazzi et al. 94]

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:







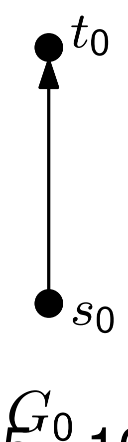
Lower Bound for the Area

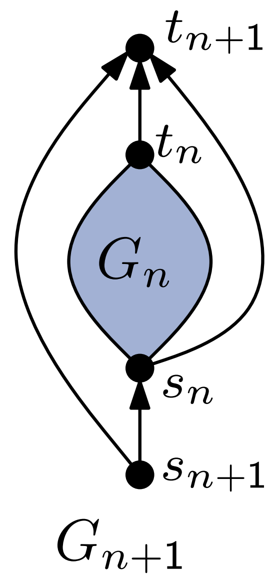
Theorem [Bertolazzi et al. 94]

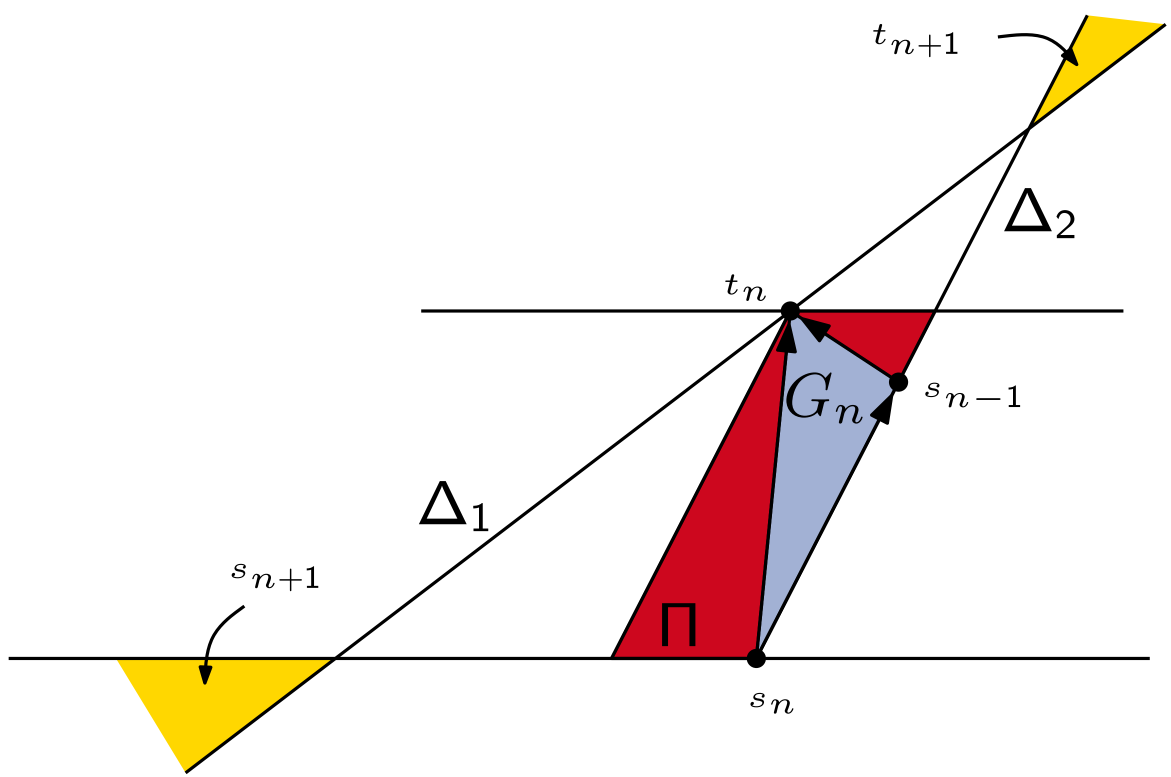
There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:

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







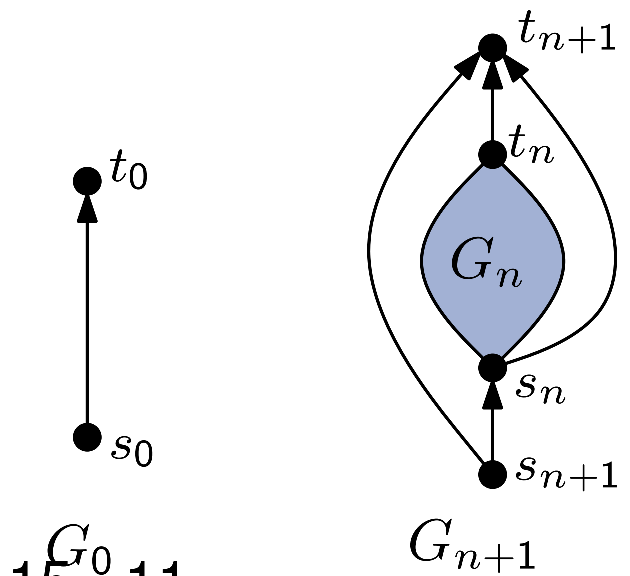
Lower Bound for the Area

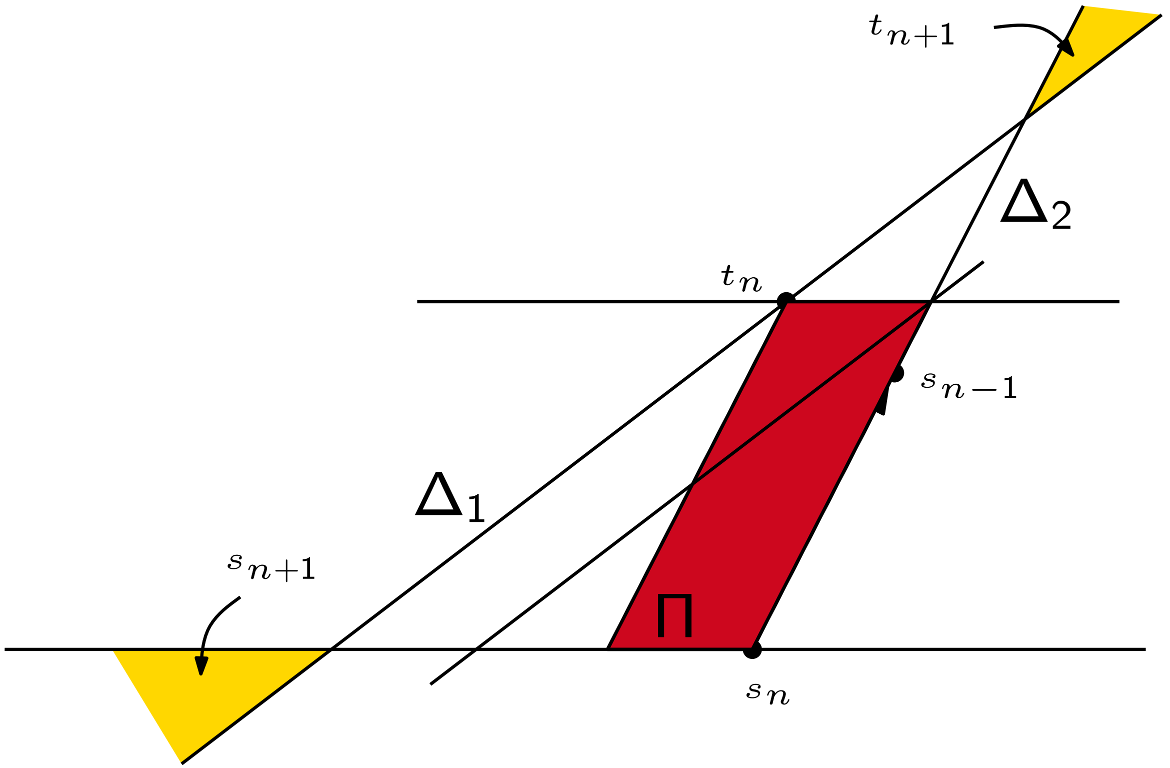
Theorem [Bertolazzi et al. 94]

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: $Area(\Pi) > 2 \cdot Area(G_n)$





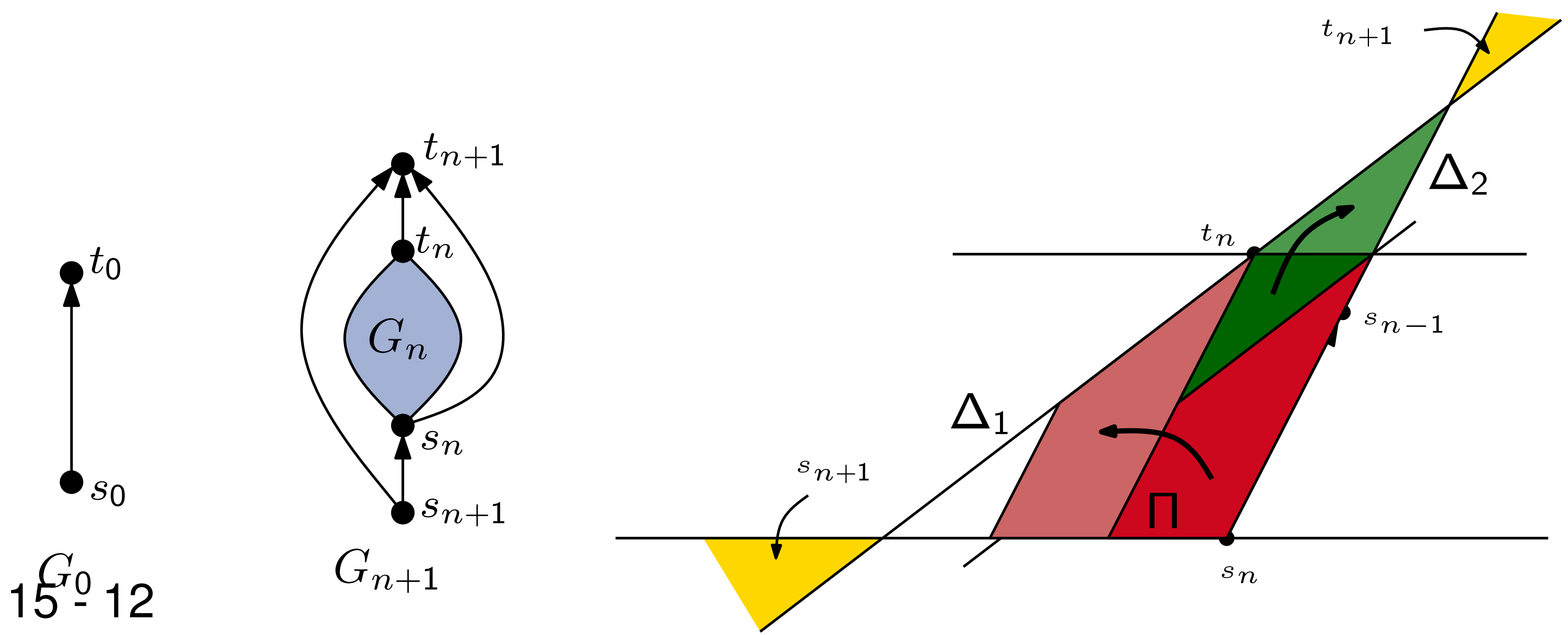
Lower Bound for the Area

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: $Area(\Pi) > 2 \cdot Area(G_n)$



15 - 12

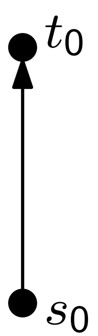
Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

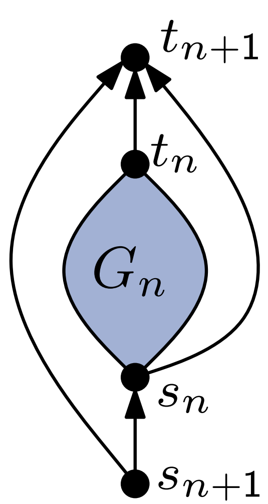
There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:

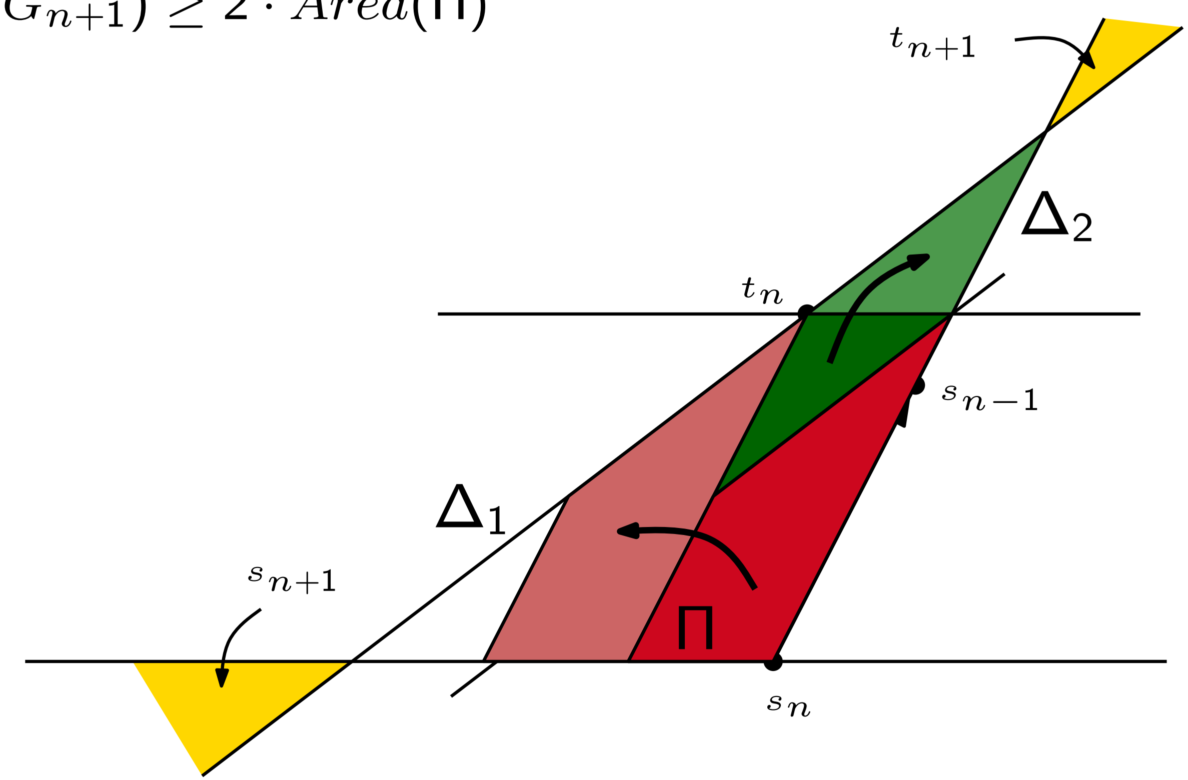
- We have that: $Area(\Pi) > 2 \cdot Area(G_n)$
- $Area(G_{n+1}) \geq 2 \cdot Area(\Pi)$



G_0



G_{n+1}



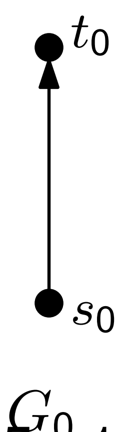
Lower Bound for the Area

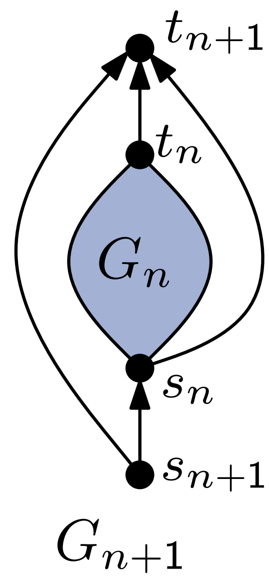
Theorem [Bertolazzi et al. 94]

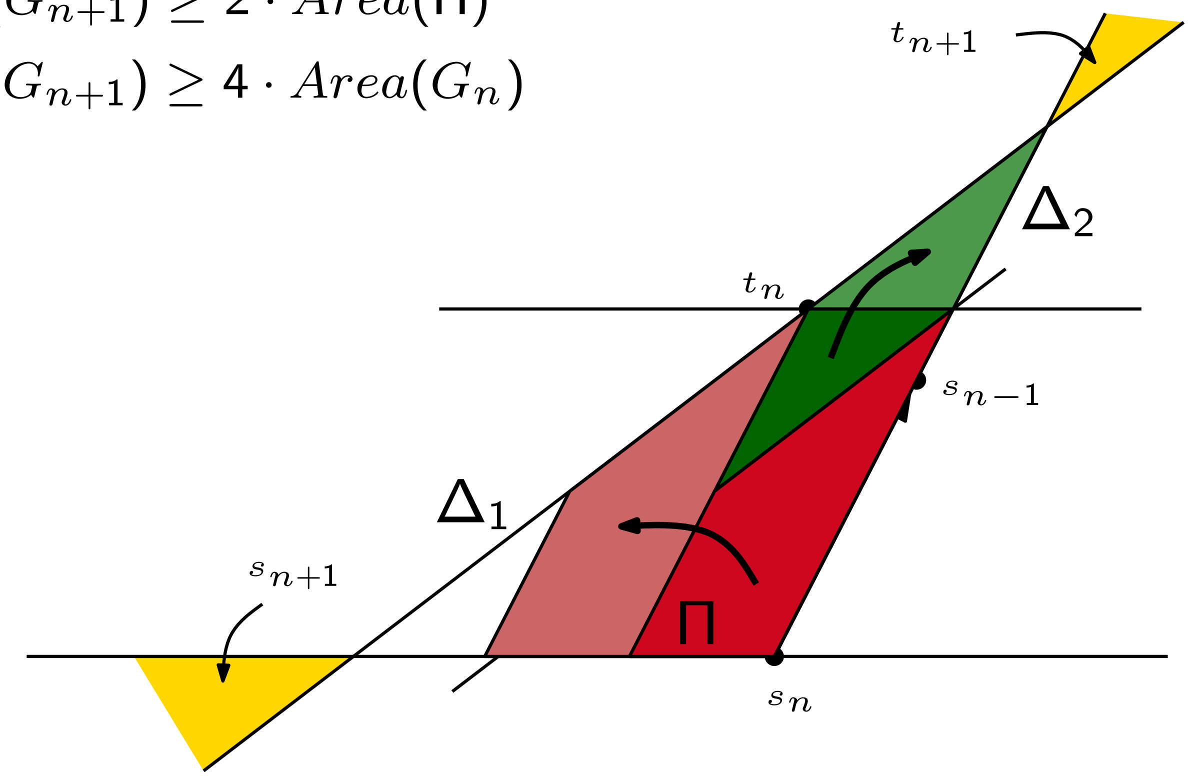
There exists a $2n$-vertex series-parallel graph G_n such that any upward planar drawing of G_n respecting embedding requires area $\Omega(4^n)$.

Proof:

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







Overview

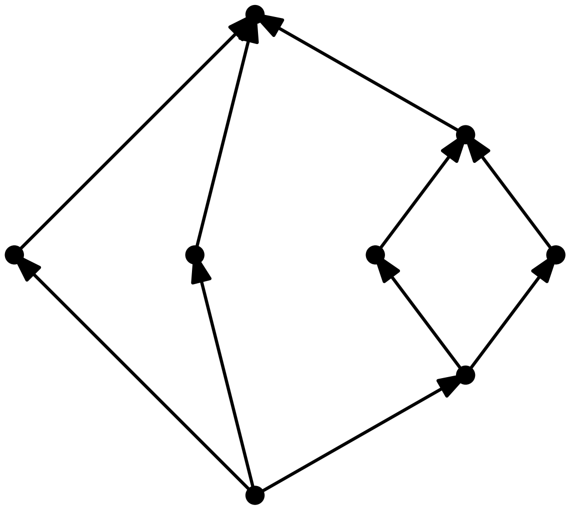
Series - parallel graph. Definition and Decomposition.

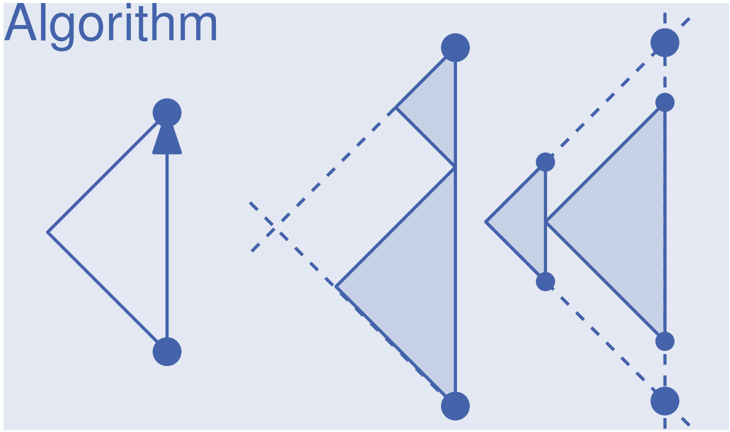
Algorithm for upward straight-line drawing.

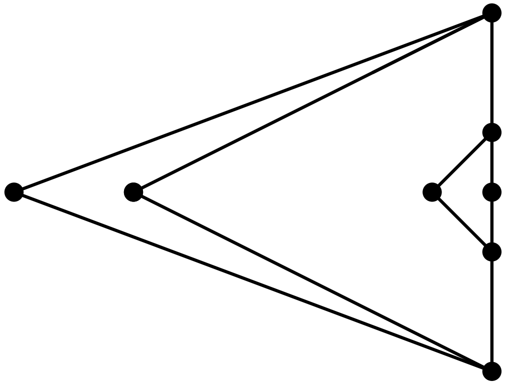
Lower bound on the area.

Symmetry display for series-parallel graphs.

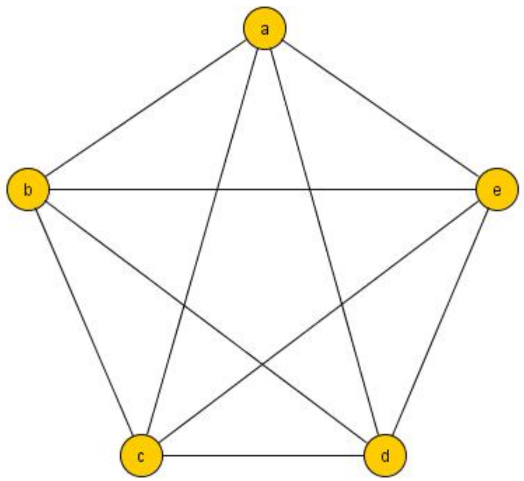
Property of the Algorithm

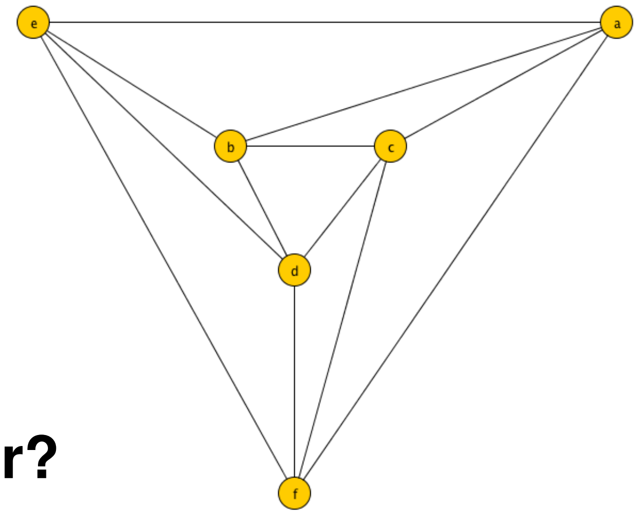




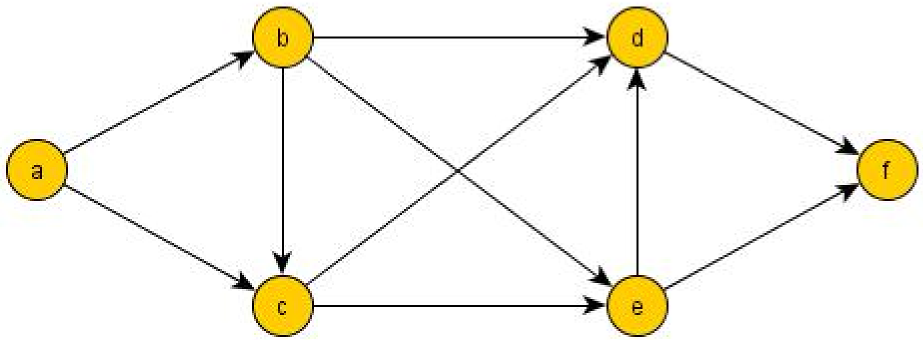


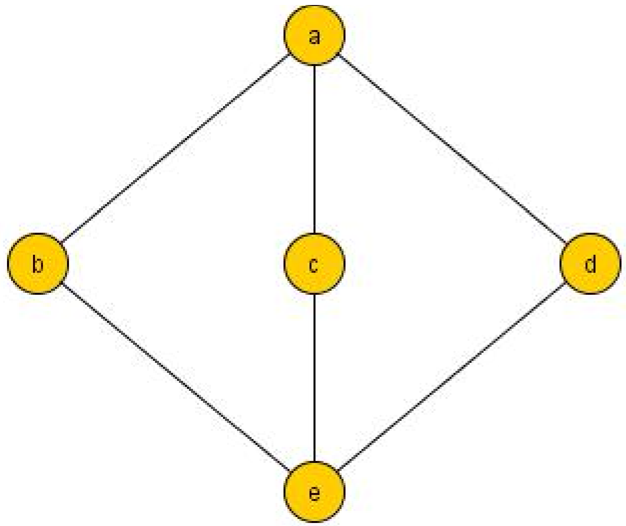
Property of the Algorithm



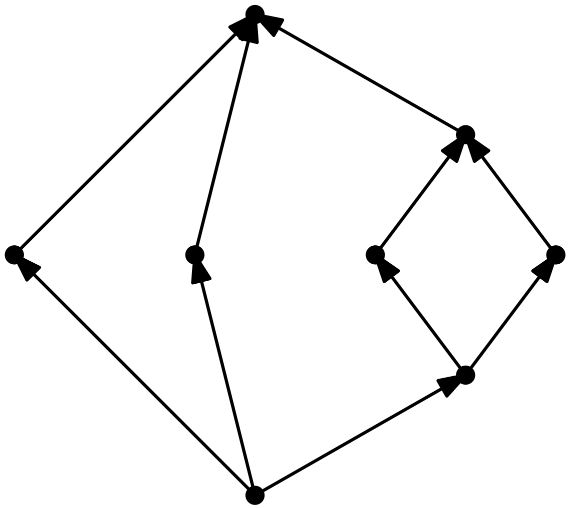


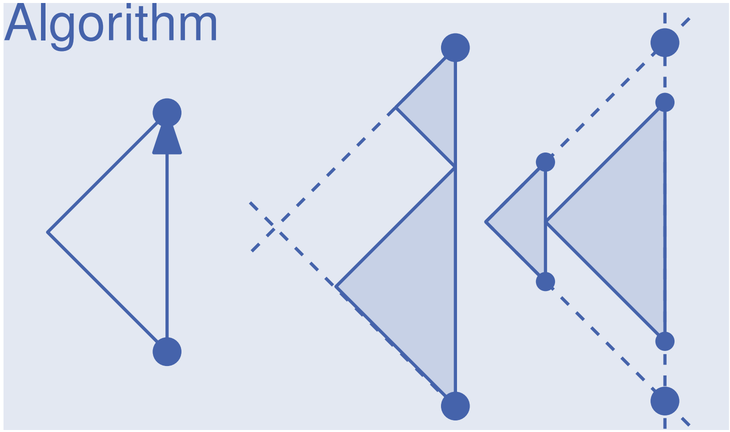
Do You Remember?

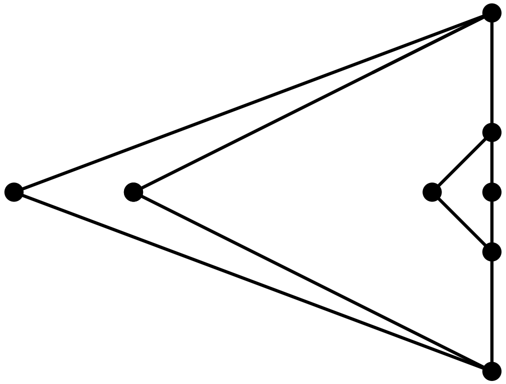




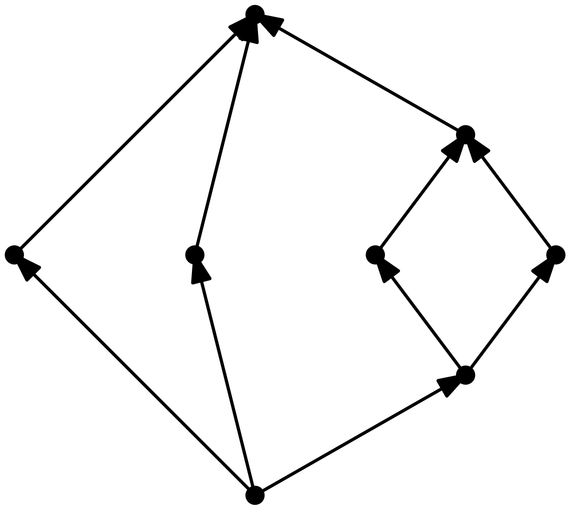
Property of the Algorithm

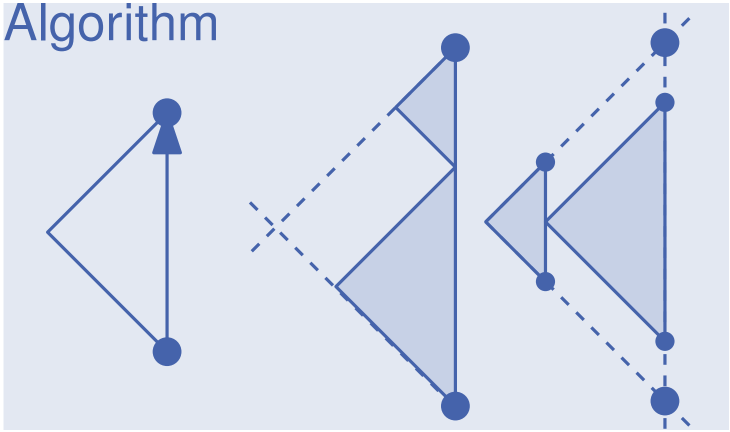


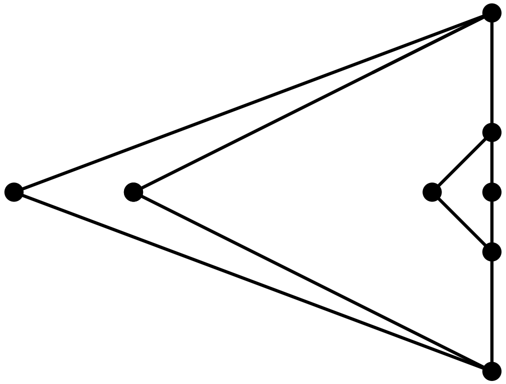


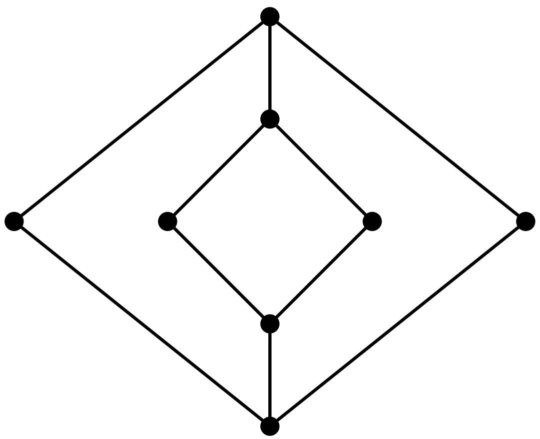


Property of the Algorithm



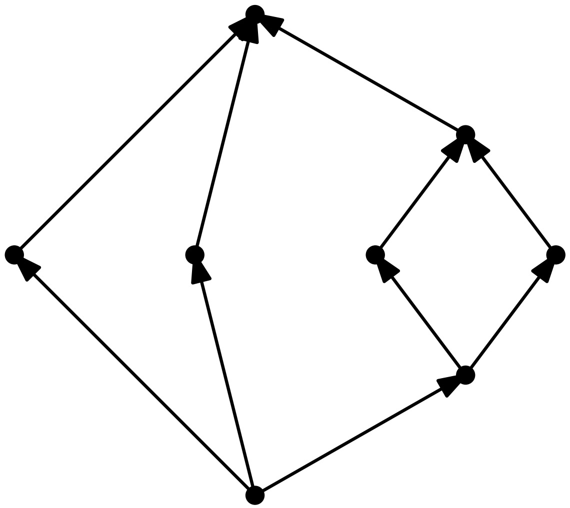


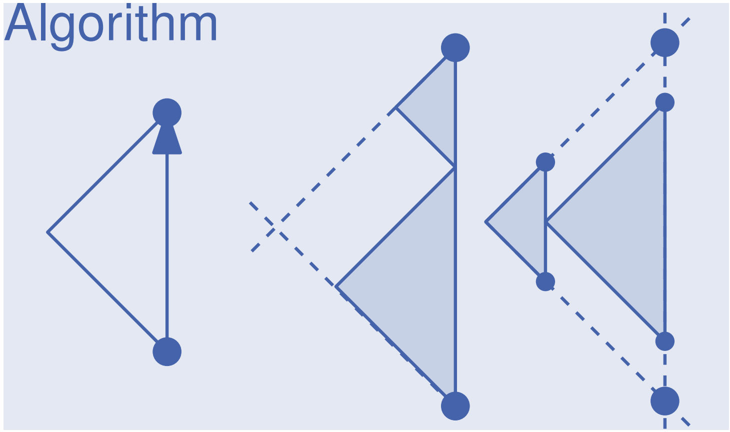


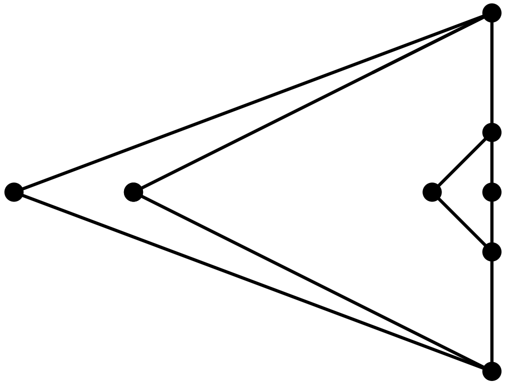


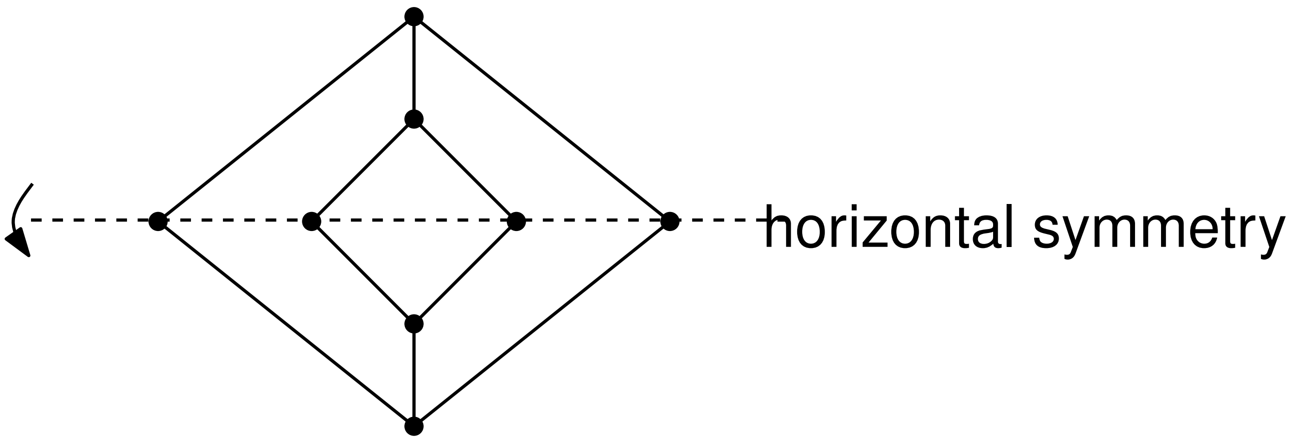
nicer???

Property of the Algorithm



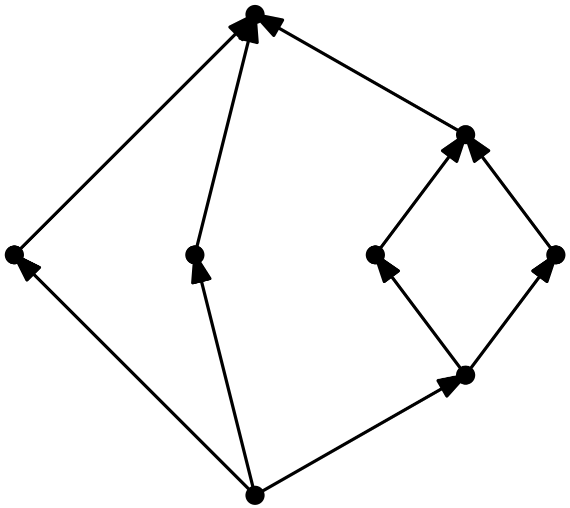


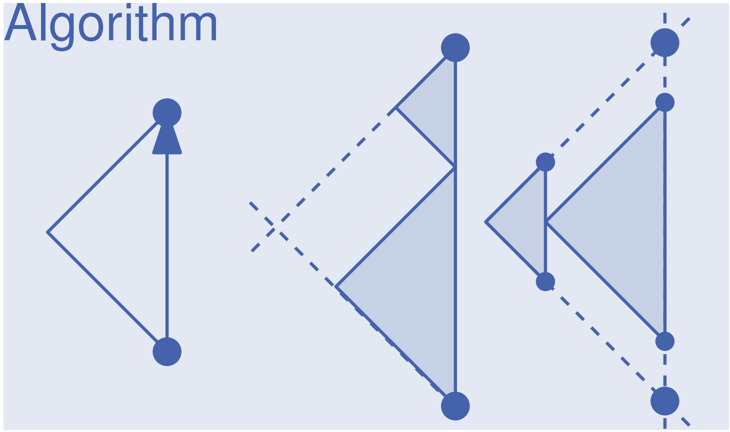


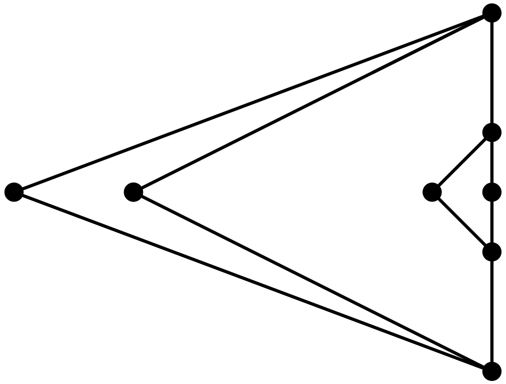


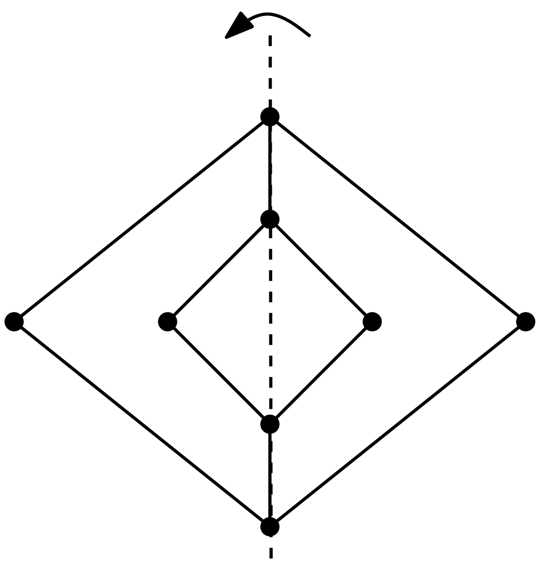
nicer???

Property of the Algorithm





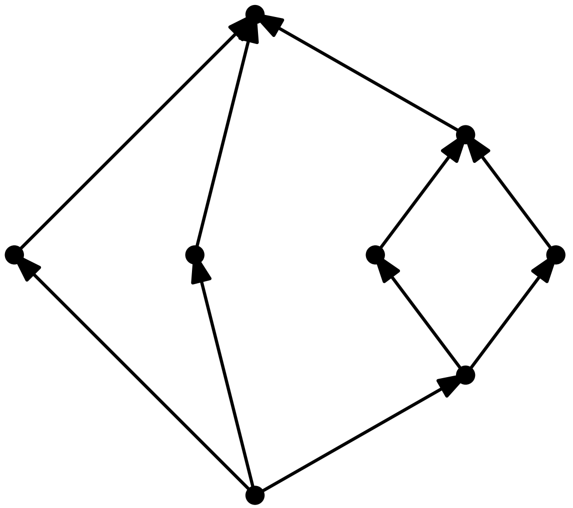


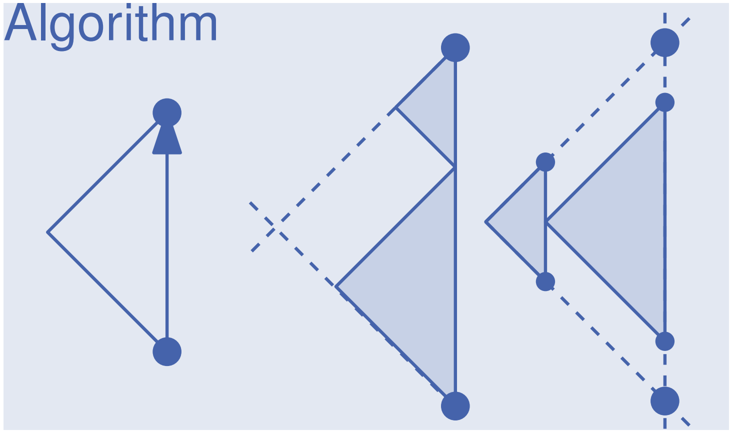


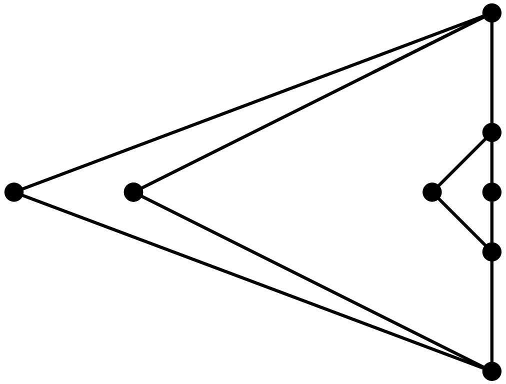
vertical symmetry

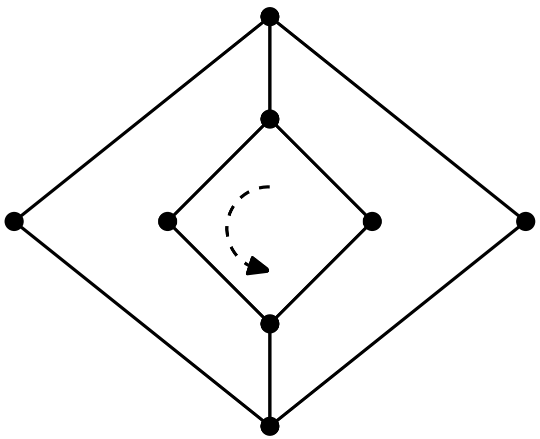
nicer???

Property of the Algorithm









rotational symmetry

nicer???

17 - 7

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

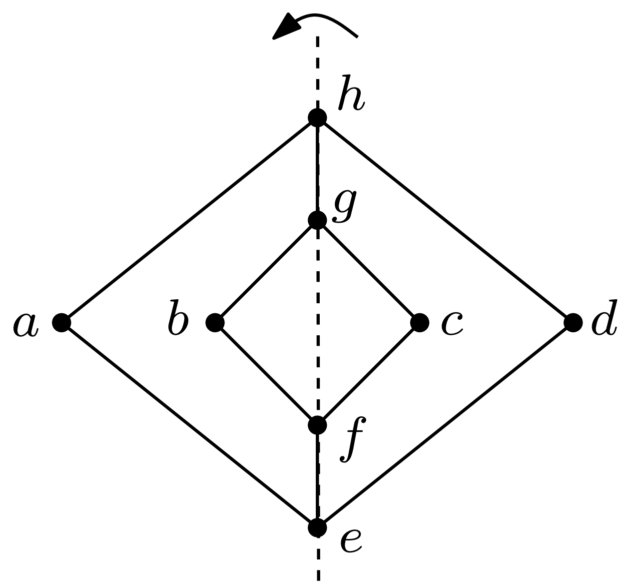
Graph Automorphism

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

Example of automorphism is a permutation of vertex set a, b, c, d, e, f, h, g defined by the rules $b \to c \to b$, $a \to d \to a$, so d, c, b, a, e, f, h, g



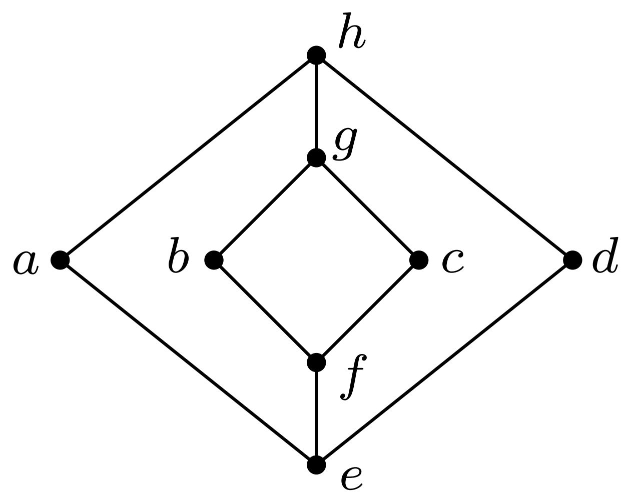
Graph Automorphism

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.



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

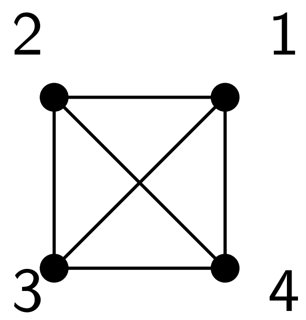
An **automorphism** of a directed graph $G = (V, E)$ is a permutation of the vertex set which preserves adjacency of the vertices and either preserves or reverses all the directions of the edges:

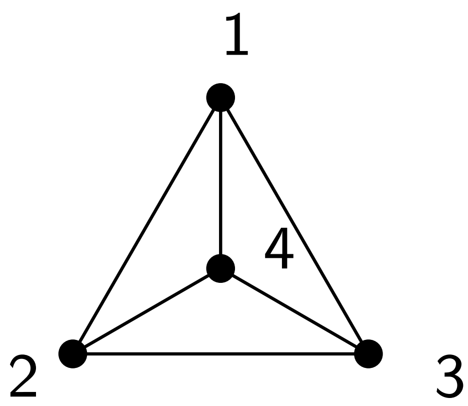
- $(u, v) \in E \Leftrightarrow (\pi(u), \pi(v)) \in E$, or
- $(u, v) \in E \Leftrightarrow (\pi(v), \pi(u)) \in E$

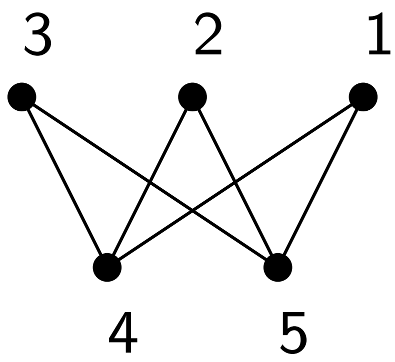
- The set of all automorphisms (direction preserving and reversing) forms the **automorphism group** of G.
- Finding an automorphism group of a graph is *isomorphism complete*, that is equivalent to testing whether two graphs are isomorphic.
- For planar graphs, graphs with bounded degree isomorphism problem has polynomial-time algorithms (for more see citations in [HEL00]).

Geometric Automorphism

- Geometric realizability of automorphisms:

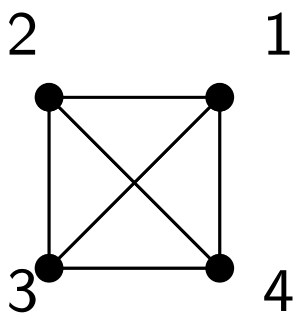


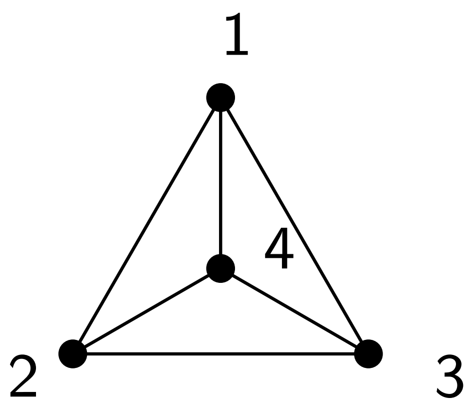


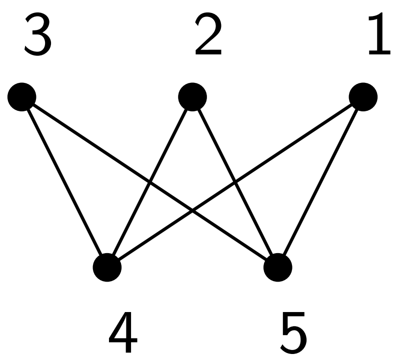


Geometric Automorphism

- Geometric realizability of automorphisms:





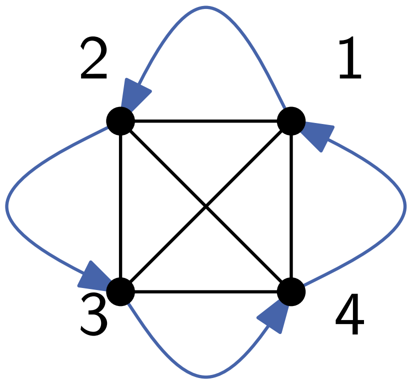


Pick one of the graphs above and try to list all its automorphisms. Which of them are represented as symmetries?

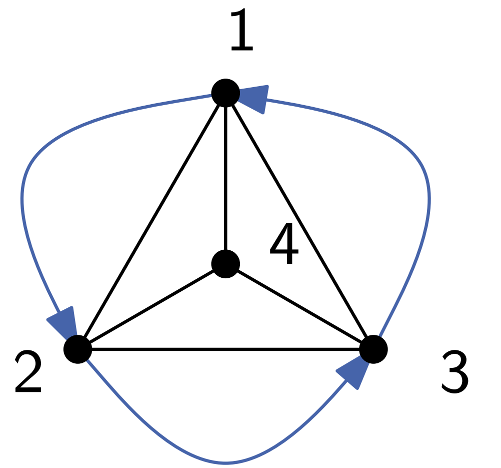
2 min

Geometric Automorphism

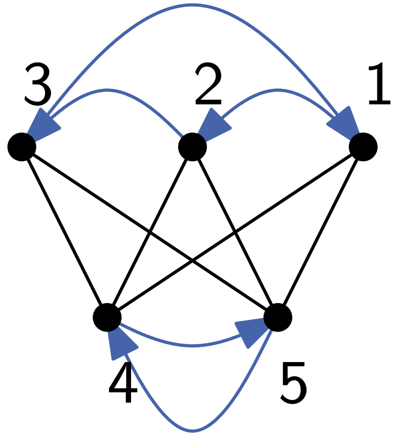
- Geometric realizability of automorphisms:



This drawing displays the automorphism $1 \to 2 \to 3 \to 4 \to 1$ as rotational symmetry. But does not shows the $1 \to 2 \to 3 \to 1$

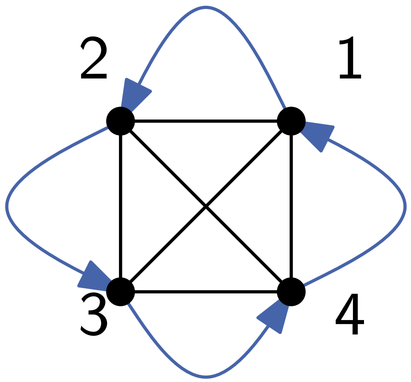


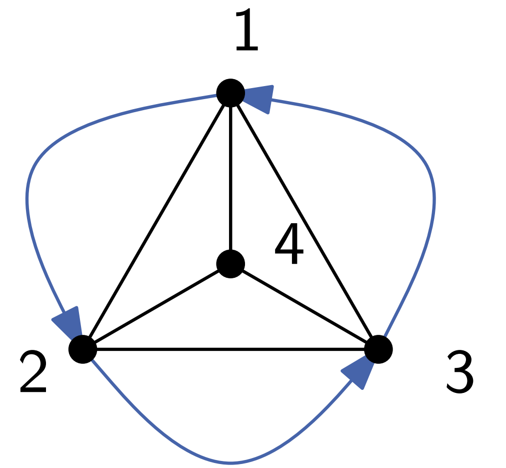
This drawing displays the automorphism $1 \to 2 \to 3 \to 1$ as rotational symmetry but not the $1 \to 2 \to 3 \to 4 \to 1$

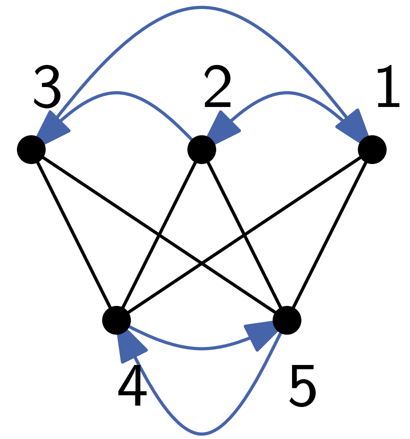


Geometric Automorphism

- Geometric realizability of automorphisms:







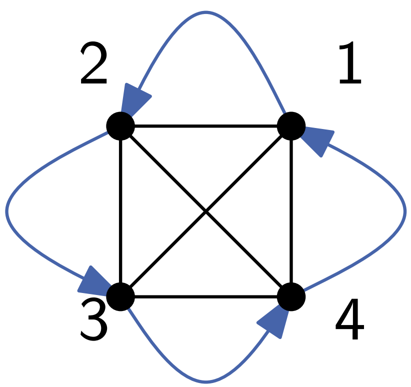
This drawing displays the automorphism $1 \to 2 \to 3 \to 4 \to 1$ as rotational symmetry. But does not shows the $1 \to 2 \to 3 \to 1$

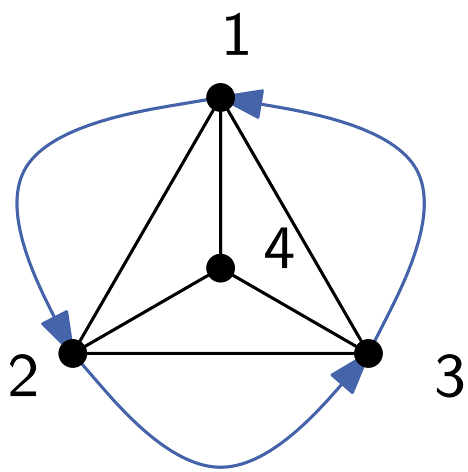
This drawing displays the automorphism $1 \to 2 \to 3 \to 1$ as rotational symmetry but not the $1 \to 2 \to 3 \to 4 \to 1$

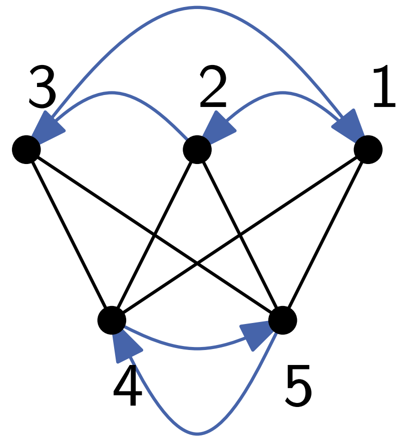
Automorphisms $1 \to 2 \to 3 \to 4 \to 1$ and $1 \to 2 \to 3 \to 1$ can not be displayed simultaneously

Geometric Automorphism

- Geometric realizability of automorphisms:







This drawing displays the automorphism $1 \to 2 \to 3 \to 4 \to 1$ as rotational symmetry. But does not shows the $1 \to 2 \to 3 \to 1$

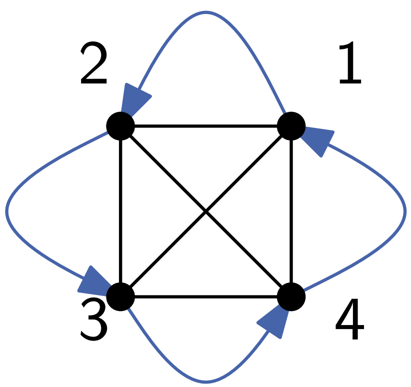
This drawing displays the automorphism $1 \to 2 \to 3 \to 1$ as rotational symmetry but not the $1 \to 2 \to 3 \to 4 \to 1$

Automorphism $1 \to 2 \to 3 \to 1$, $4 \to 5 \to 4$ is not geometrically representable. But $1 \to 3 \to 1$, $4 \to 5 \to 4$ is representable as vertical symmetry.

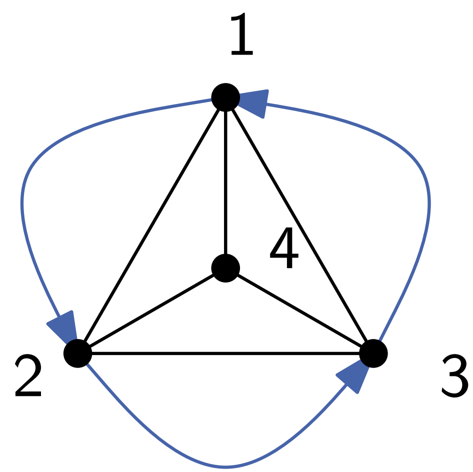
Automorphisms $1 \to 2 \to 3 \to 4 \to 1$ and $1 \to 2 \to 3 \to 1$ can not be displayed simultaneously

Geometric Automorphism

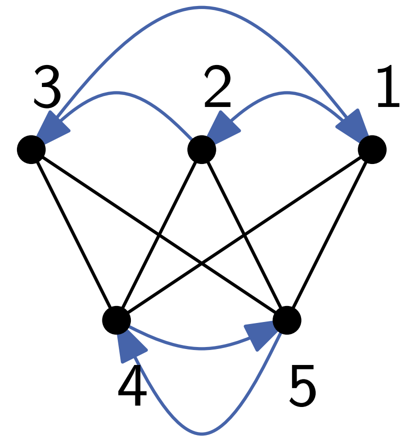
- Geometric realizability of automorphisms:



This drawing displays the automorphism $1 \to 2 \to 3 \to 4 \to 1$ as rotational symmetry. But does not shows the $1 \to 2 \to 3 \to 1$



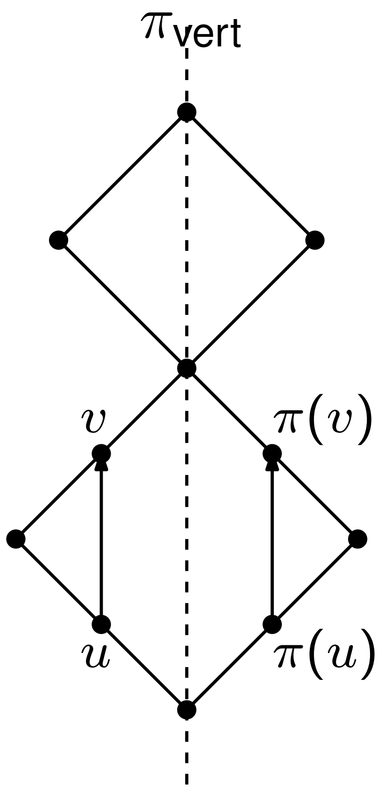
This drawing displays the automorphism $1 \to 2 \to 3 \to 1$ as rotational symmetry but not the $1 \to 2 \to 3 \to 4 \to 1$



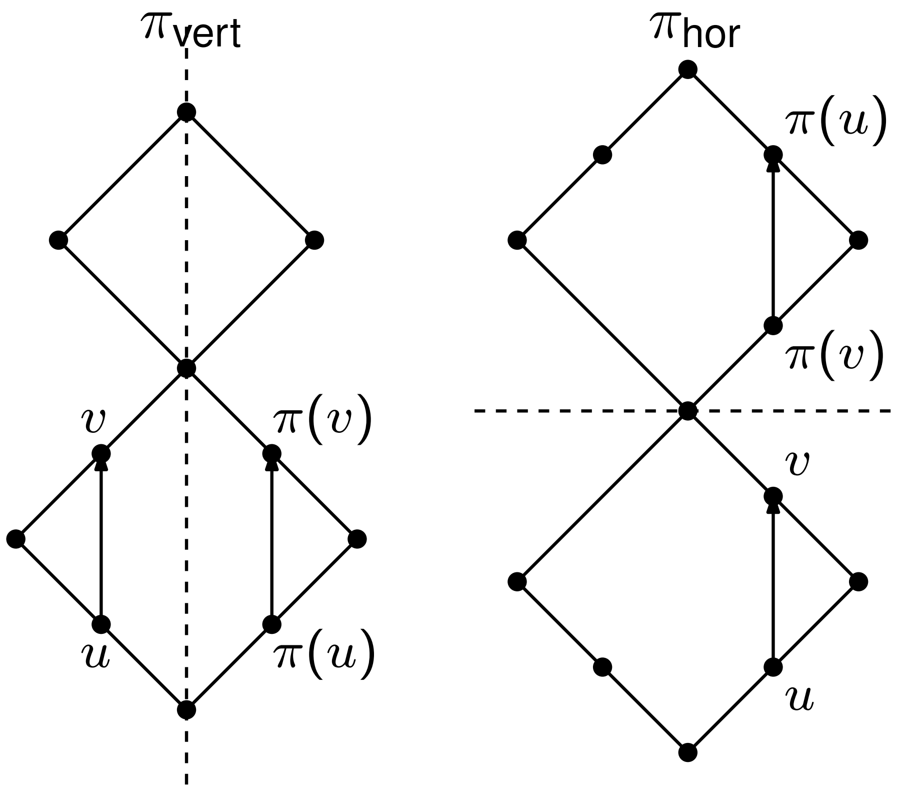
Automorphism $1 \to 2 \to 3 \to 1$, $4 \to 5 \to 4$ is not geometrically representable. But $1 \to 3 \to 1$, $4 \to 5 \to 4$ is representable as vertical symmetry.

- 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.

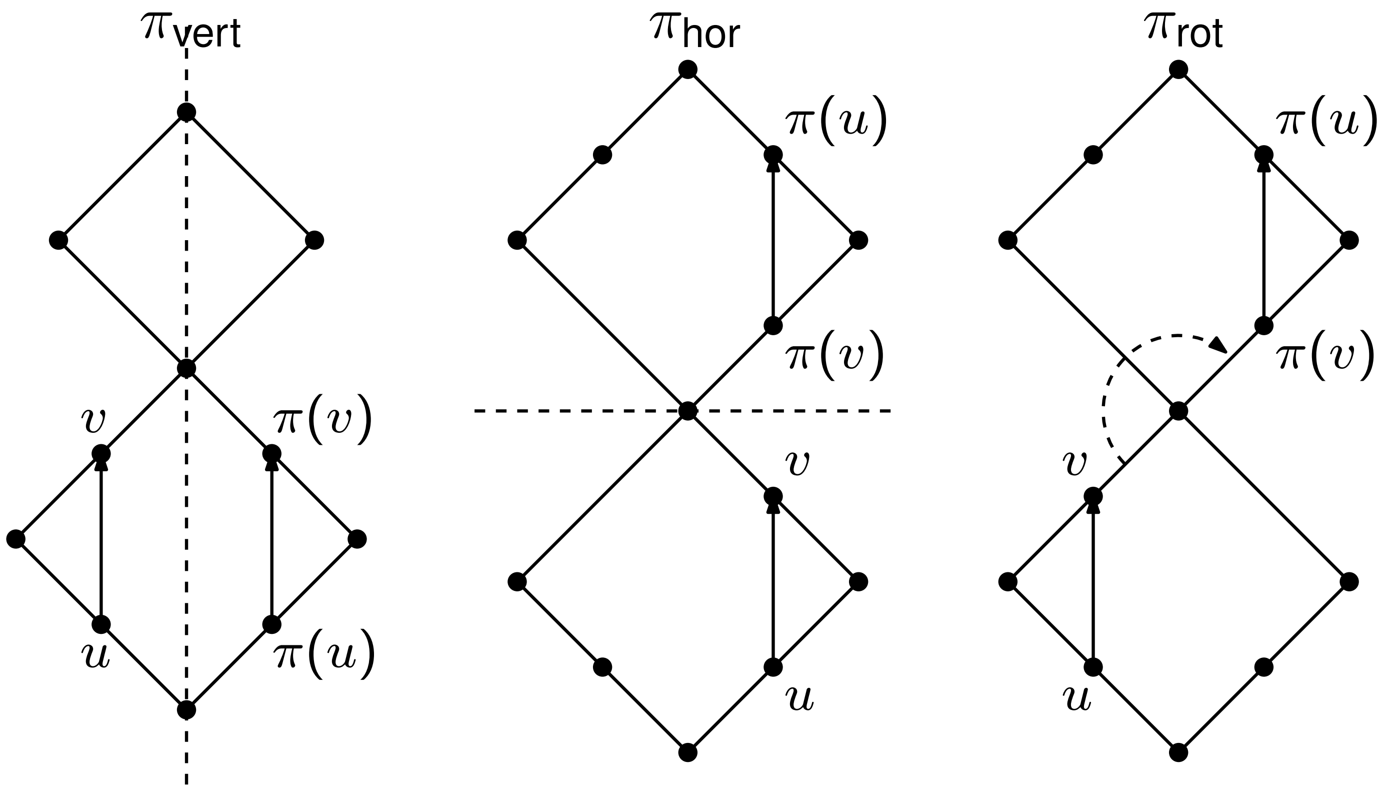
Symmetries in SP-Graphs



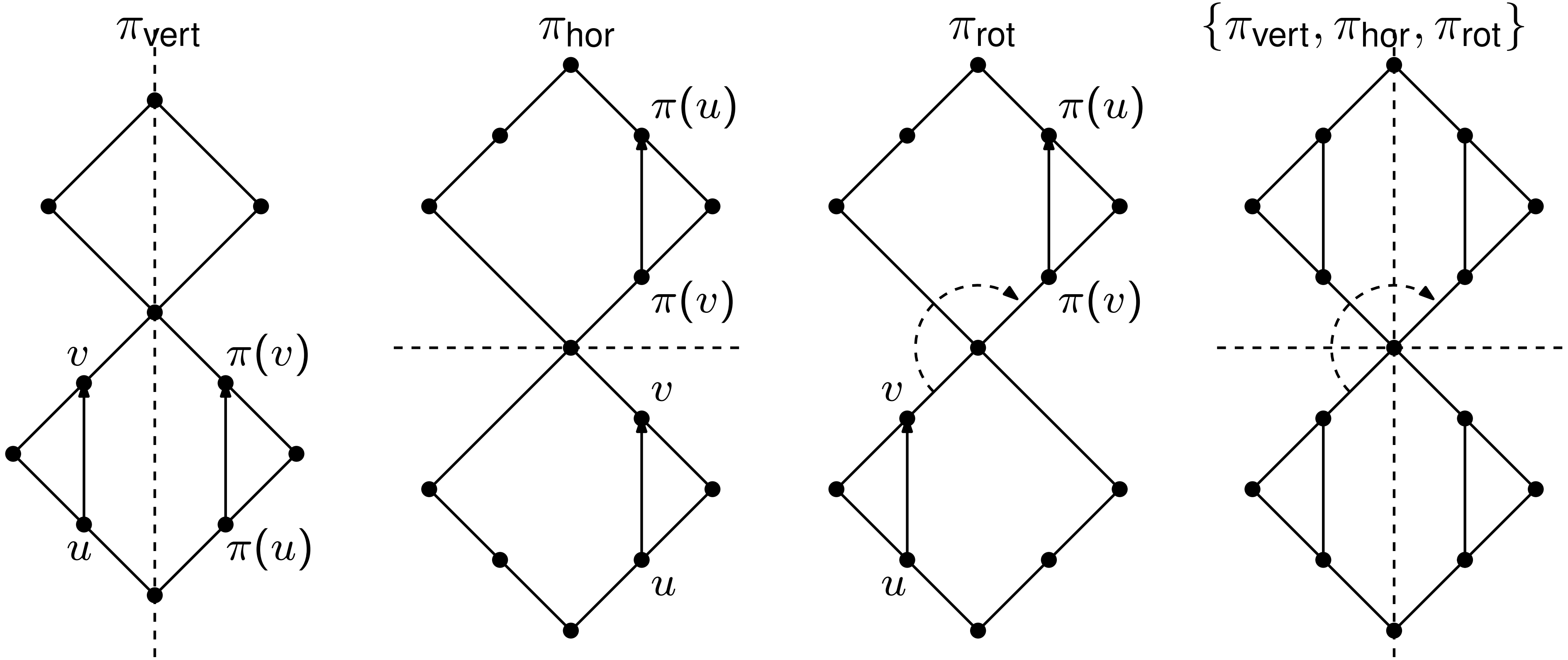
Symmetries in SP-Graphs



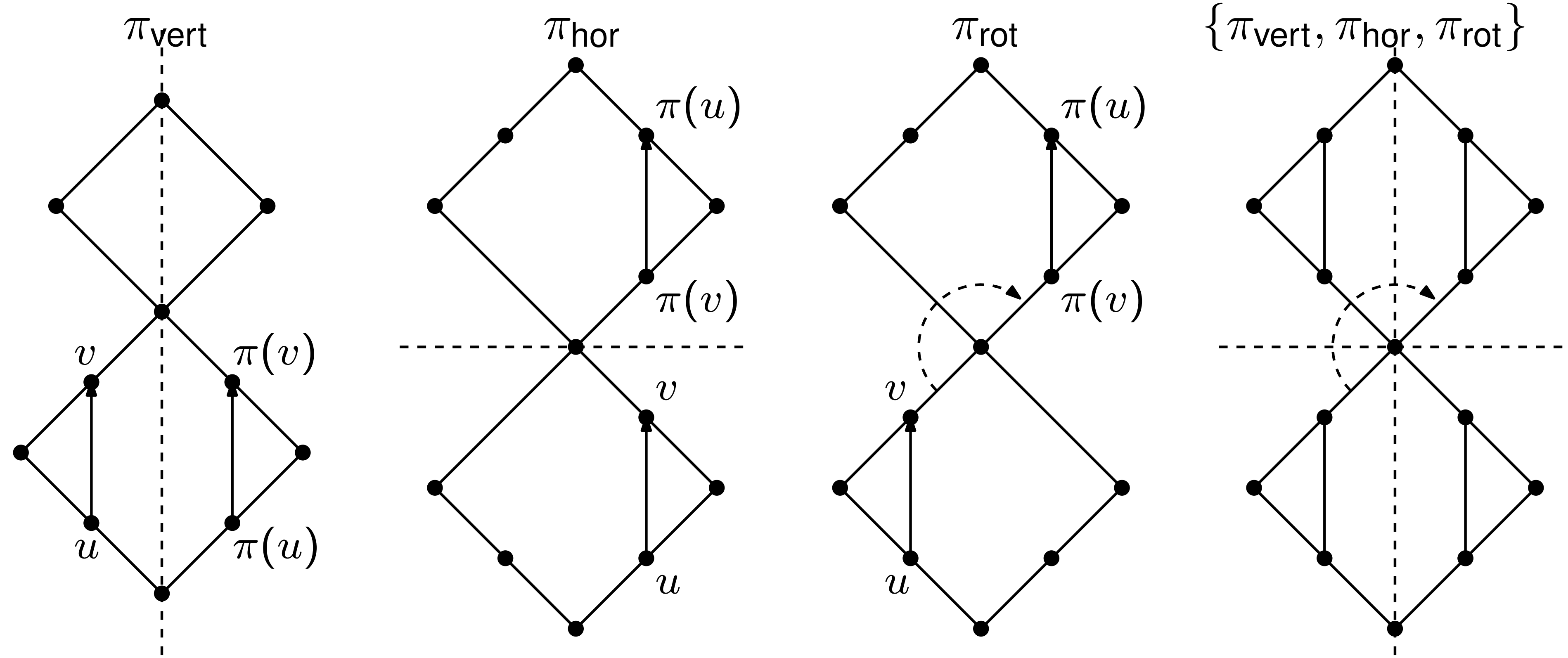
Symmetries in SP-Graphs



Symmetries in SP-Graphs

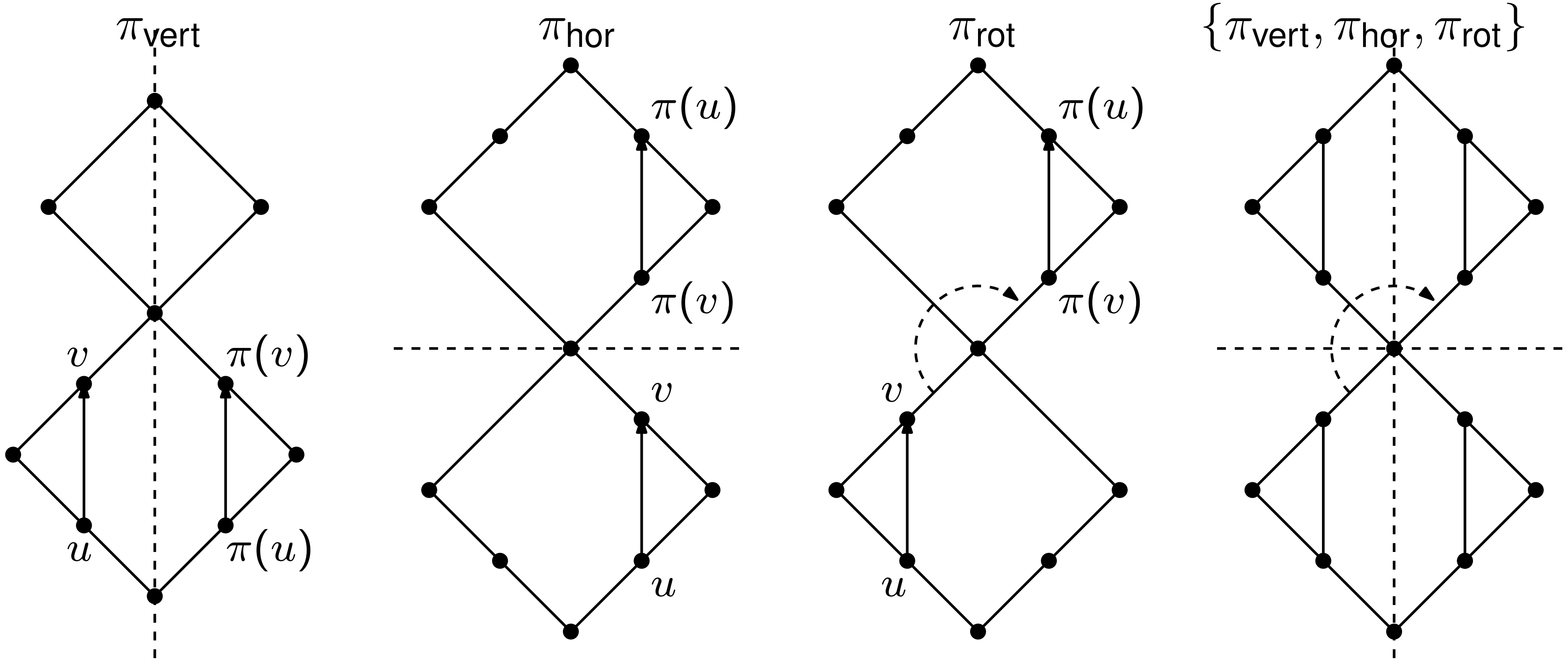


Symmetries in SP-Graphs



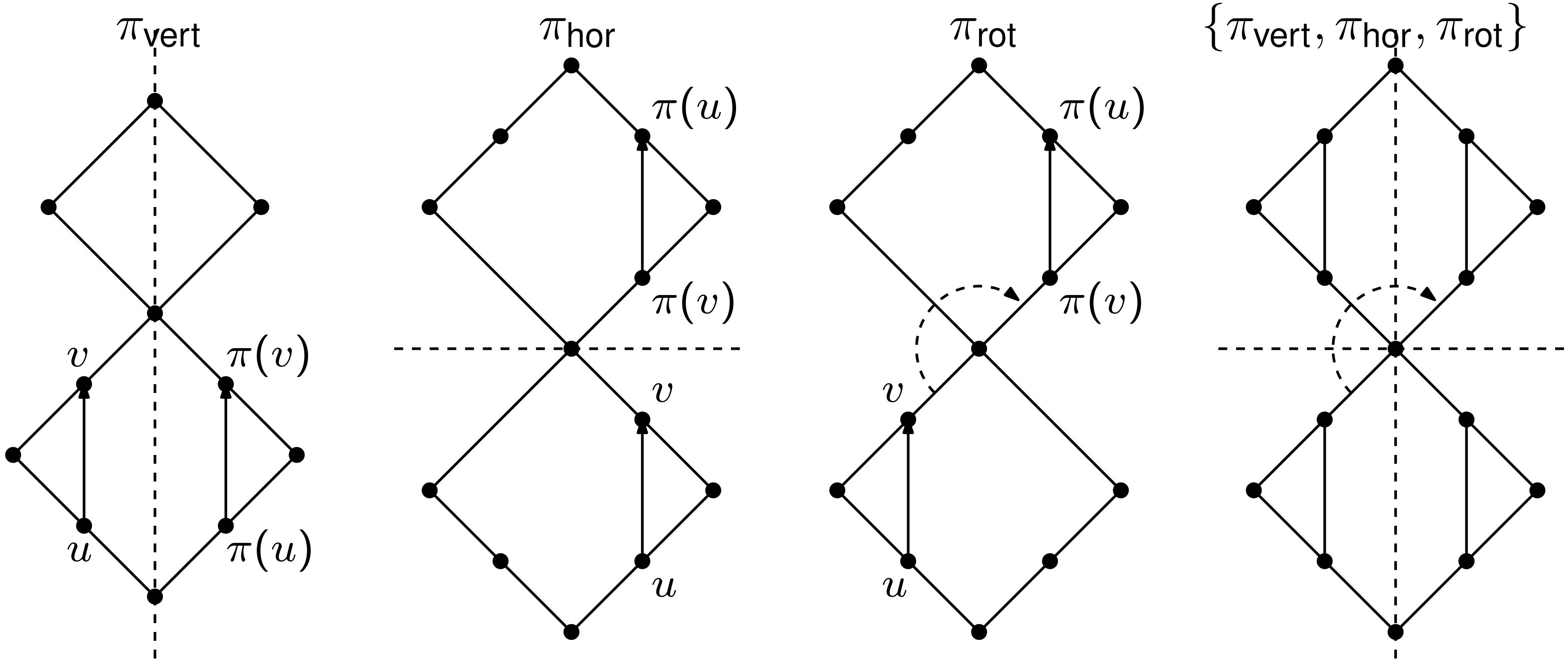
- A geometric automorphism group P of a graph G is upward planar, if there exists an upward planar drawing of G that displays each element of P as a symmetry.

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

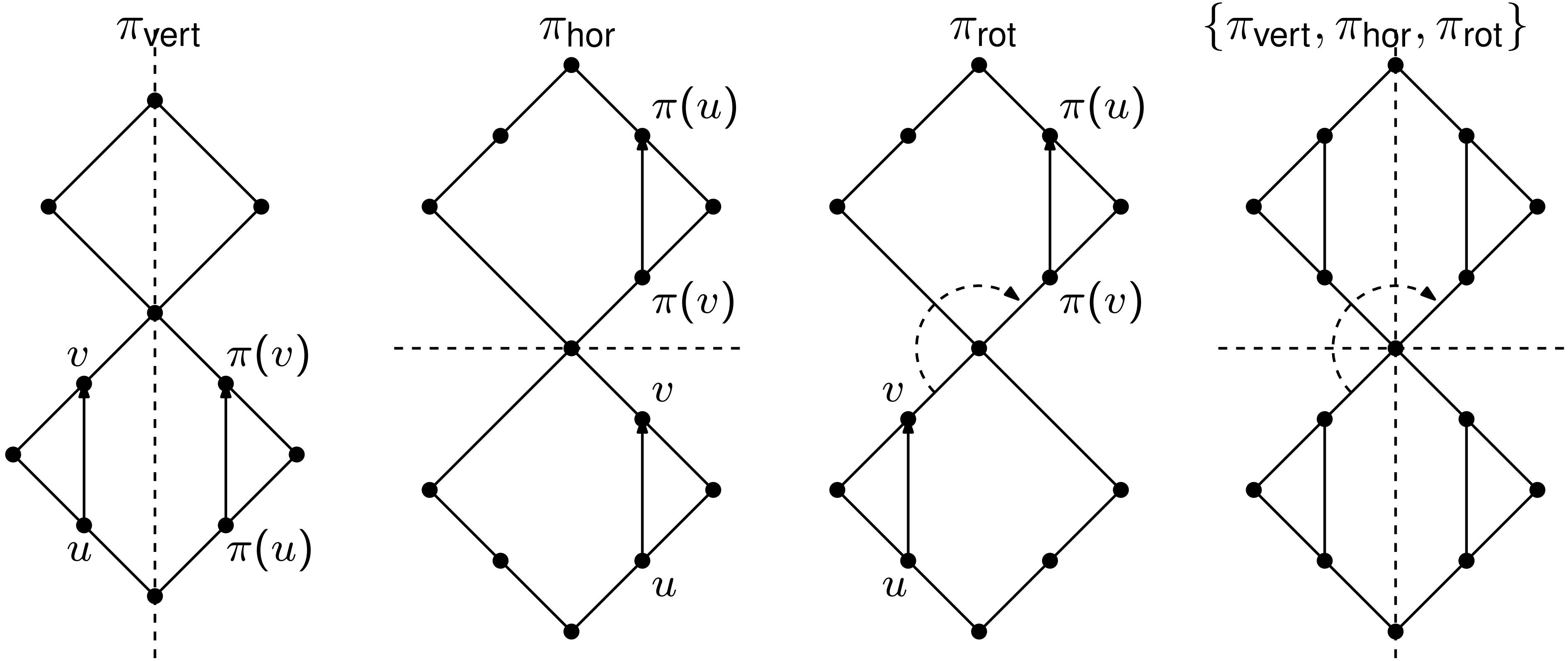


Theorem (Hong, Eades, Lee ’00) [HEL00]

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

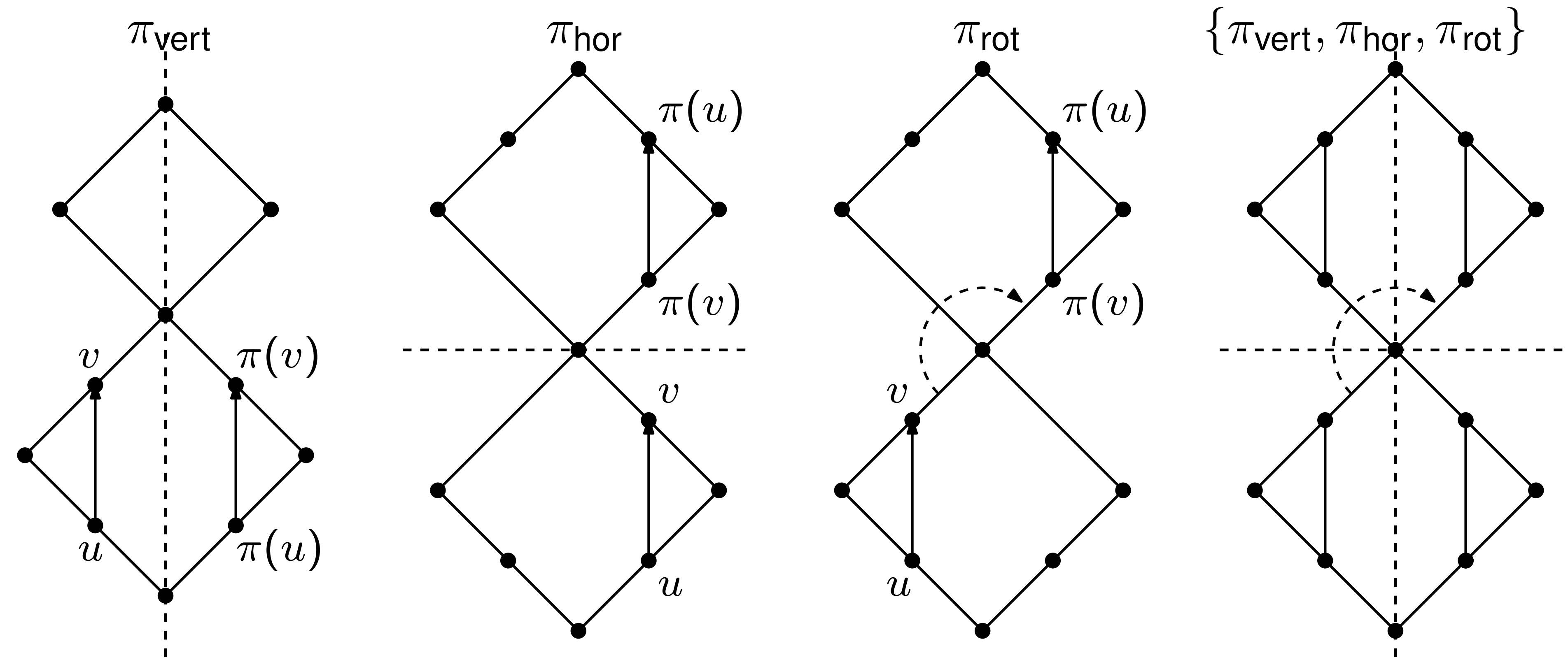
- $\{\text{id}\}$
- $\{\text{id}, \pi\}$ with $\pi \in \{\pi_{\text{vert}}, \pi_{\text{hor}}, \pi_{\text{rot}}\}$
- $\{\text{id}, \pi_{\text{vert}}, \pi_{\text{hor}}, \pi_{\text{rot}}\}$.

Symmetries in SP-Graphs



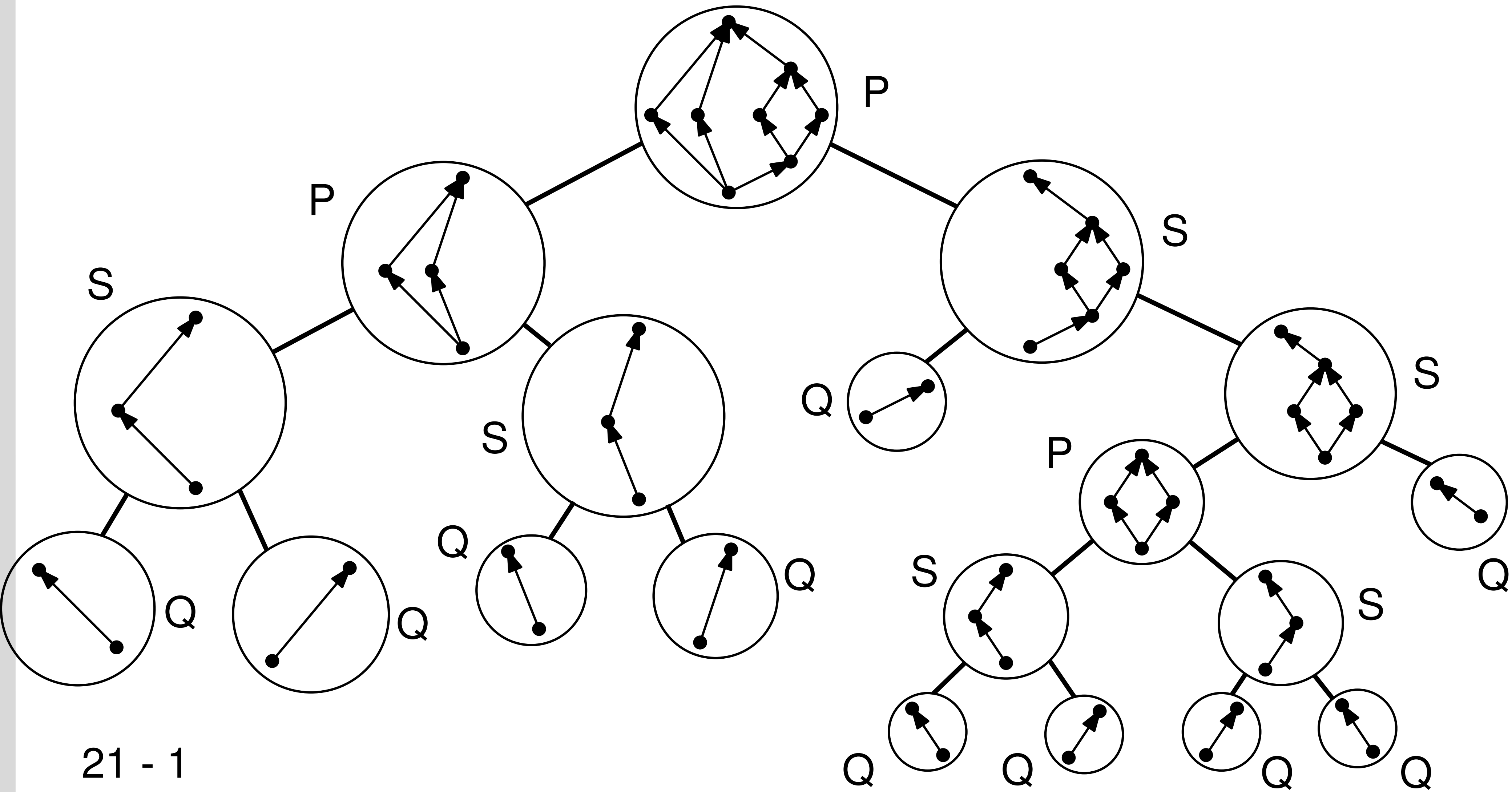
- The automorphism group of maximum size can be found in linear time.

Symmetries in SP-Graphs



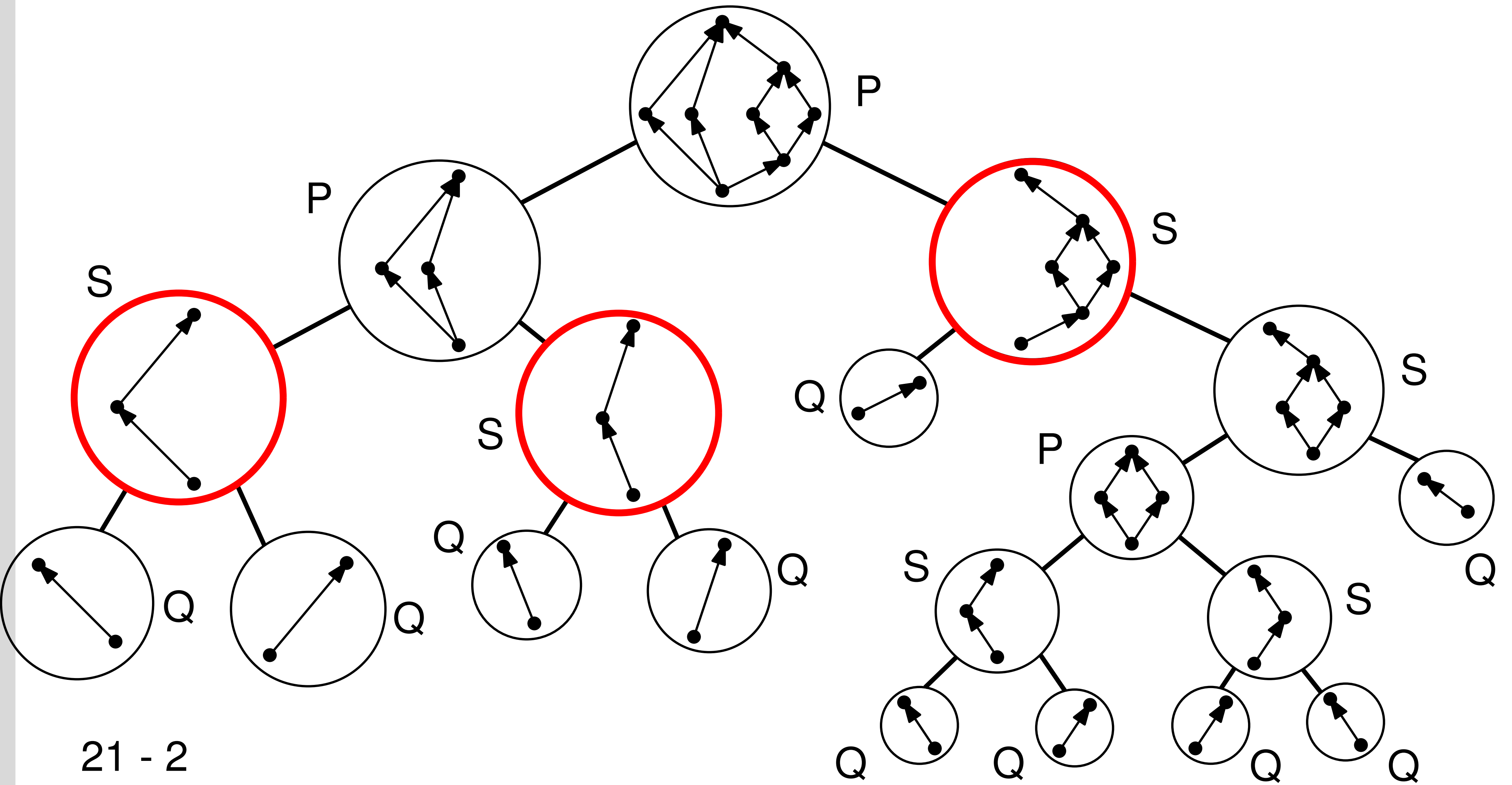
- The automorphism group of maximum size can be found in linear time.
- Given a maximum size automorphism group of a series-parallel graph, a polyline upward planar drawing that displays this automorphism can be constructed in linear time as well.

Vertical Automorphism



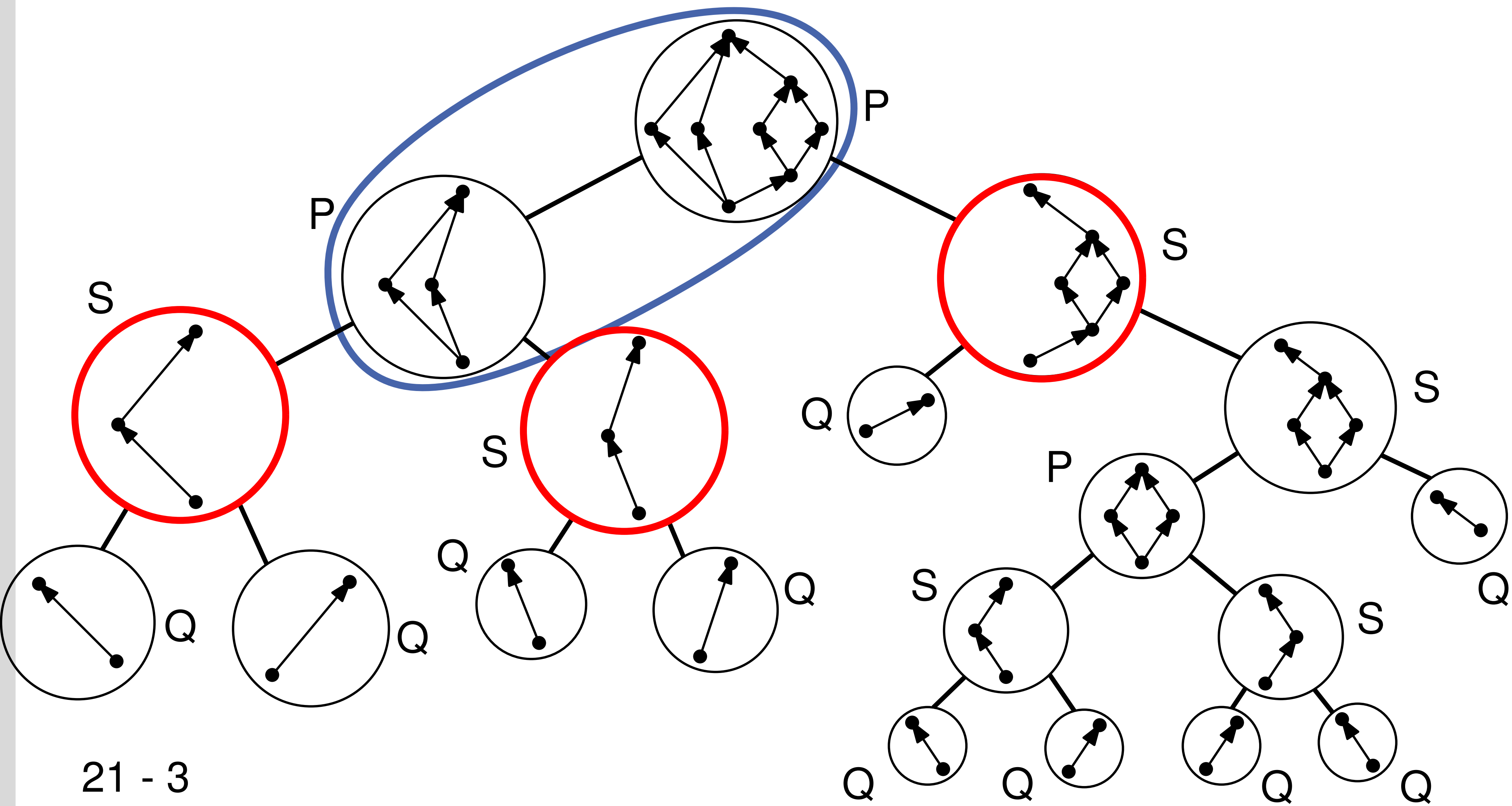
21 - 1

Vertical Automorphism



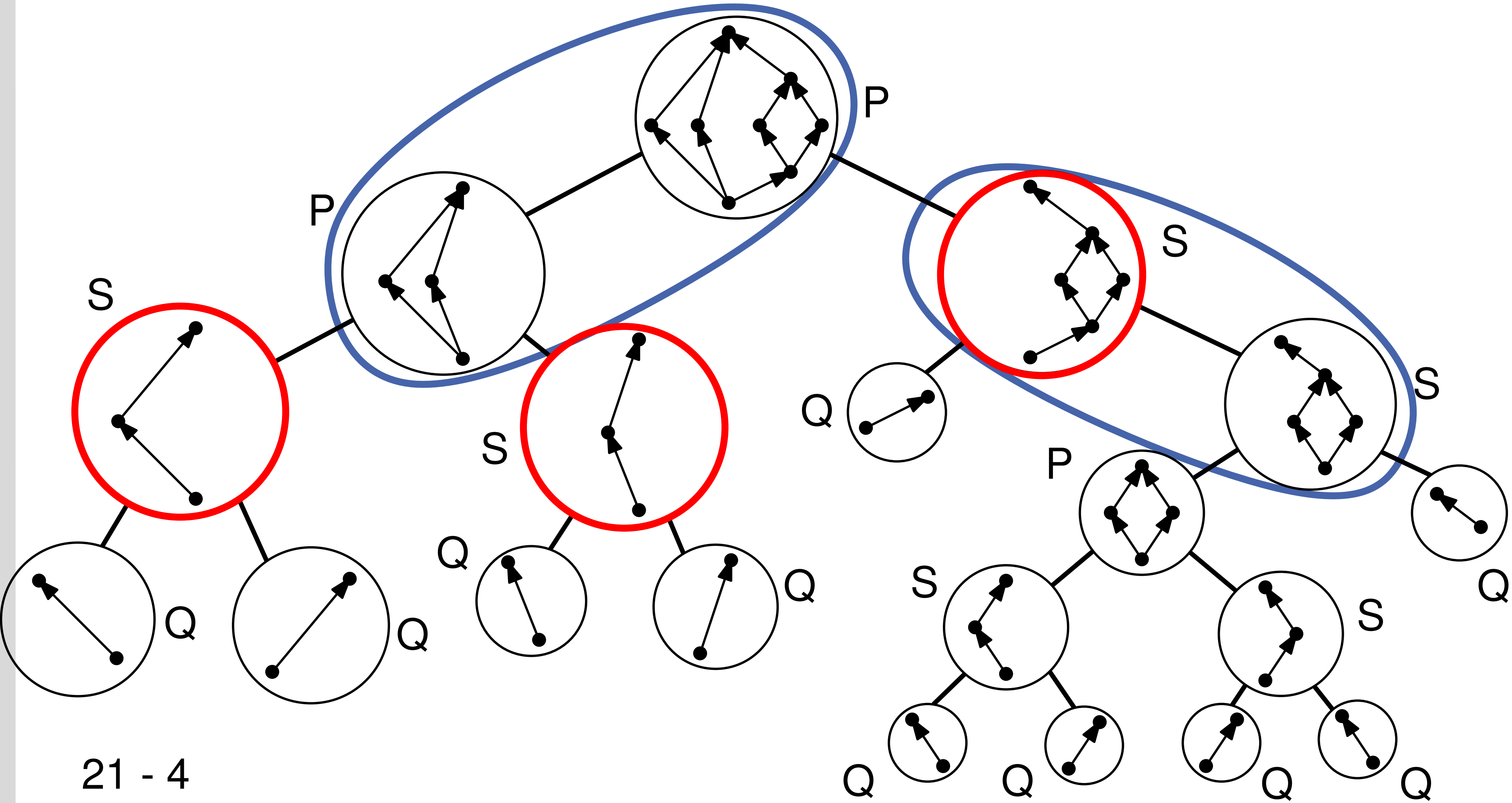
21 - 2

Vertical Automorphism

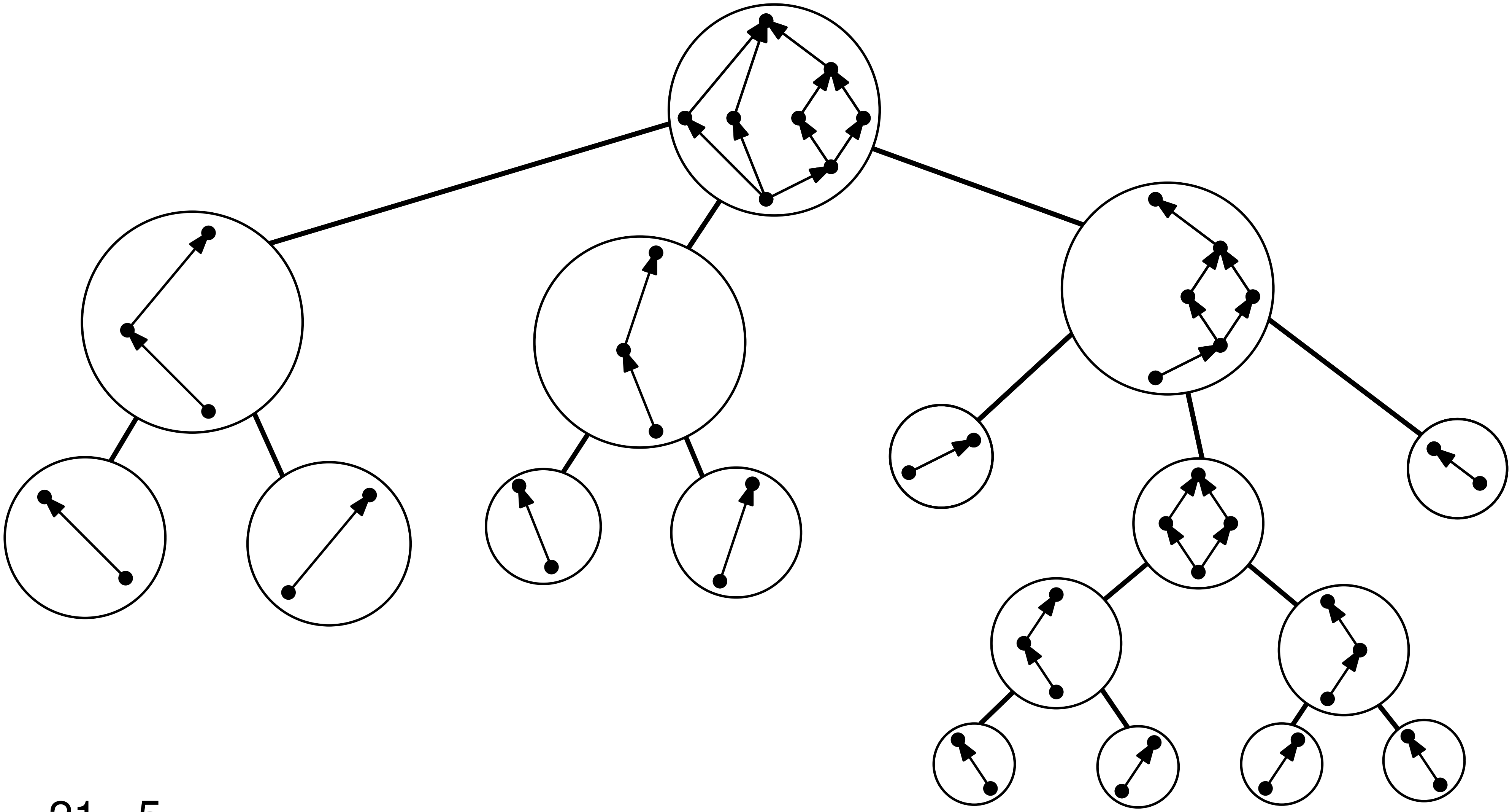


21 - 3

Vertical Automorphism



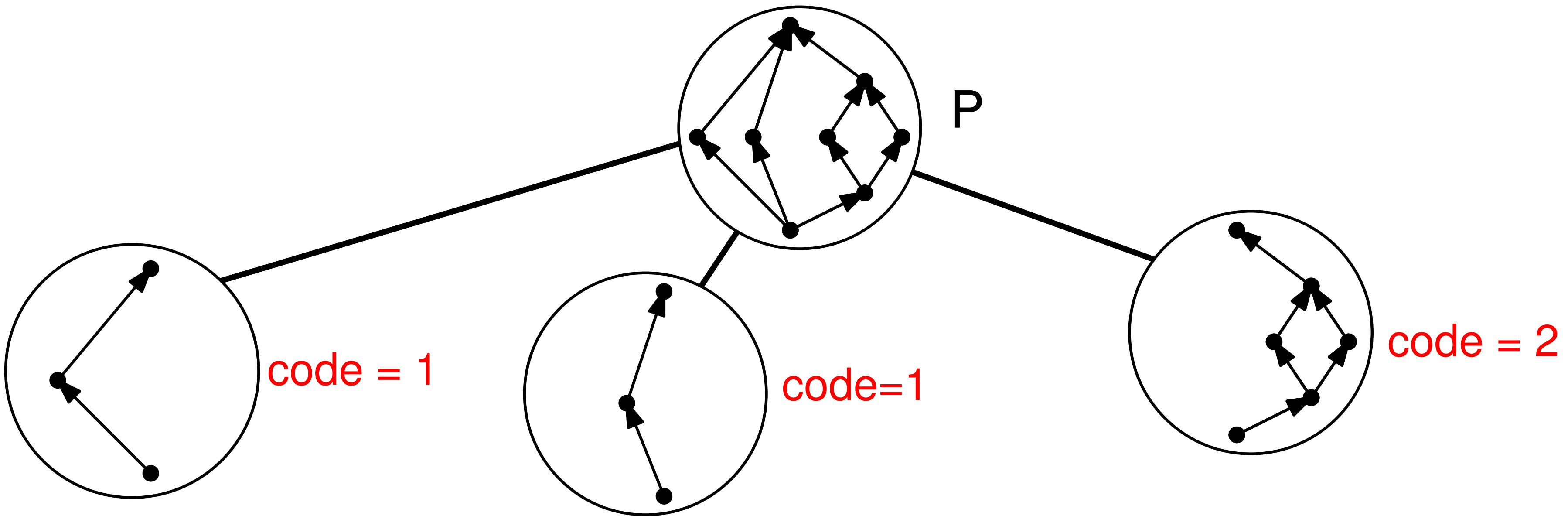
Vertical Automorphism



21 - 5

Vertical Automorphism

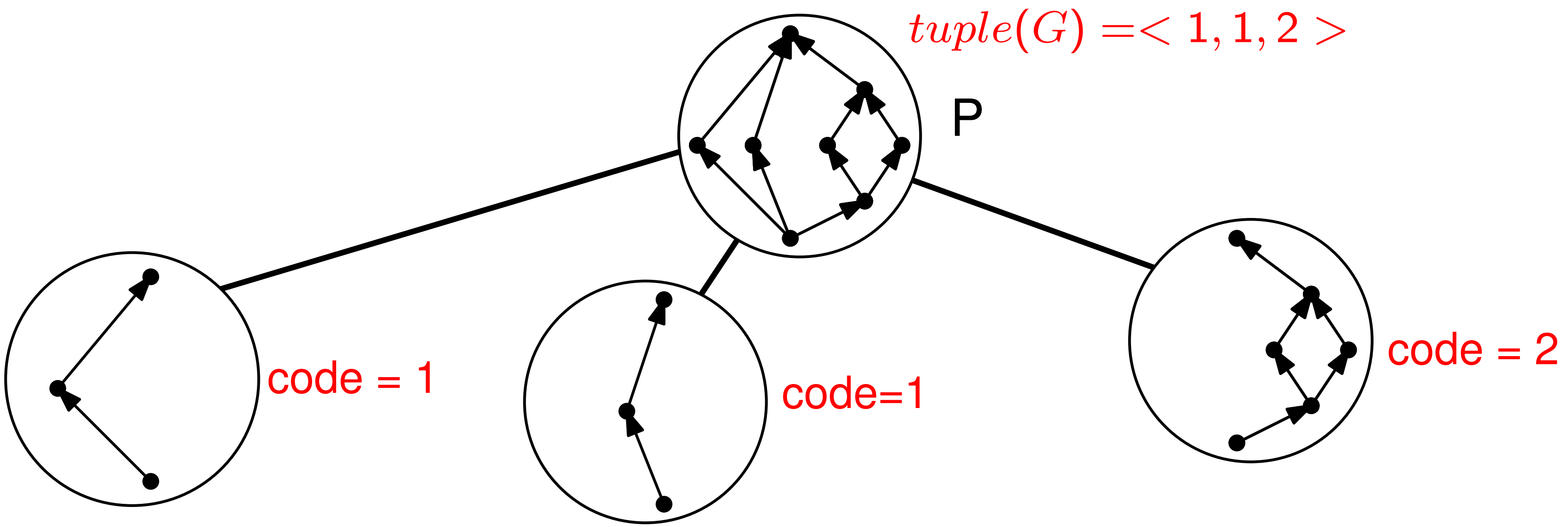
- $code(G)$ - two graphs at the same level have the same code iff they are isomorphic
- $tuple(G)$ - set of the codes of the children (sorted for a P-node)



21 - 6

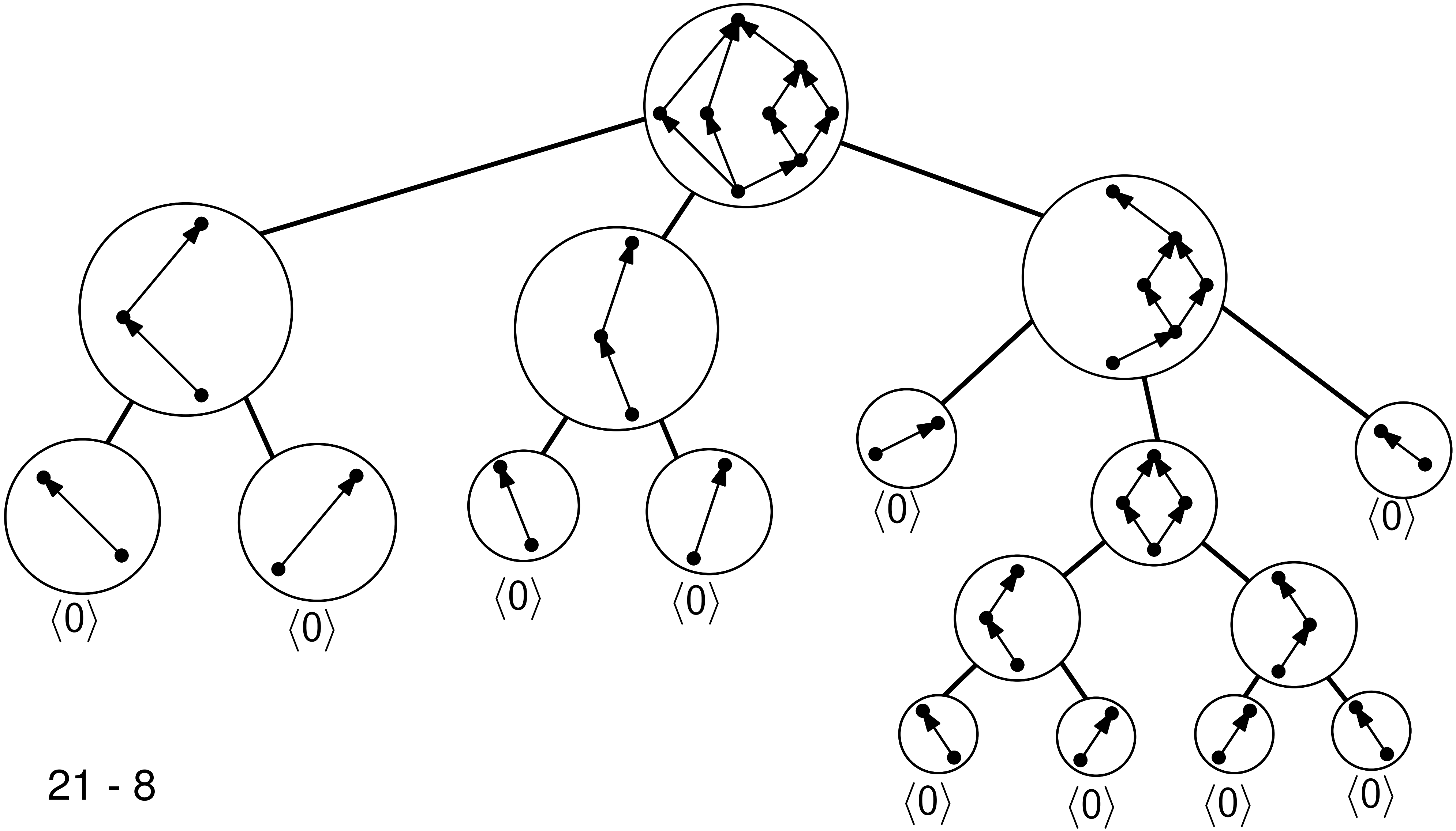
Vertical Automorphism

- $code(G)$ - two graphs at the same level have the same code iff they are isomorphic
- $tuple(G)$ - set of the codes of the children (sorted for a P-node)



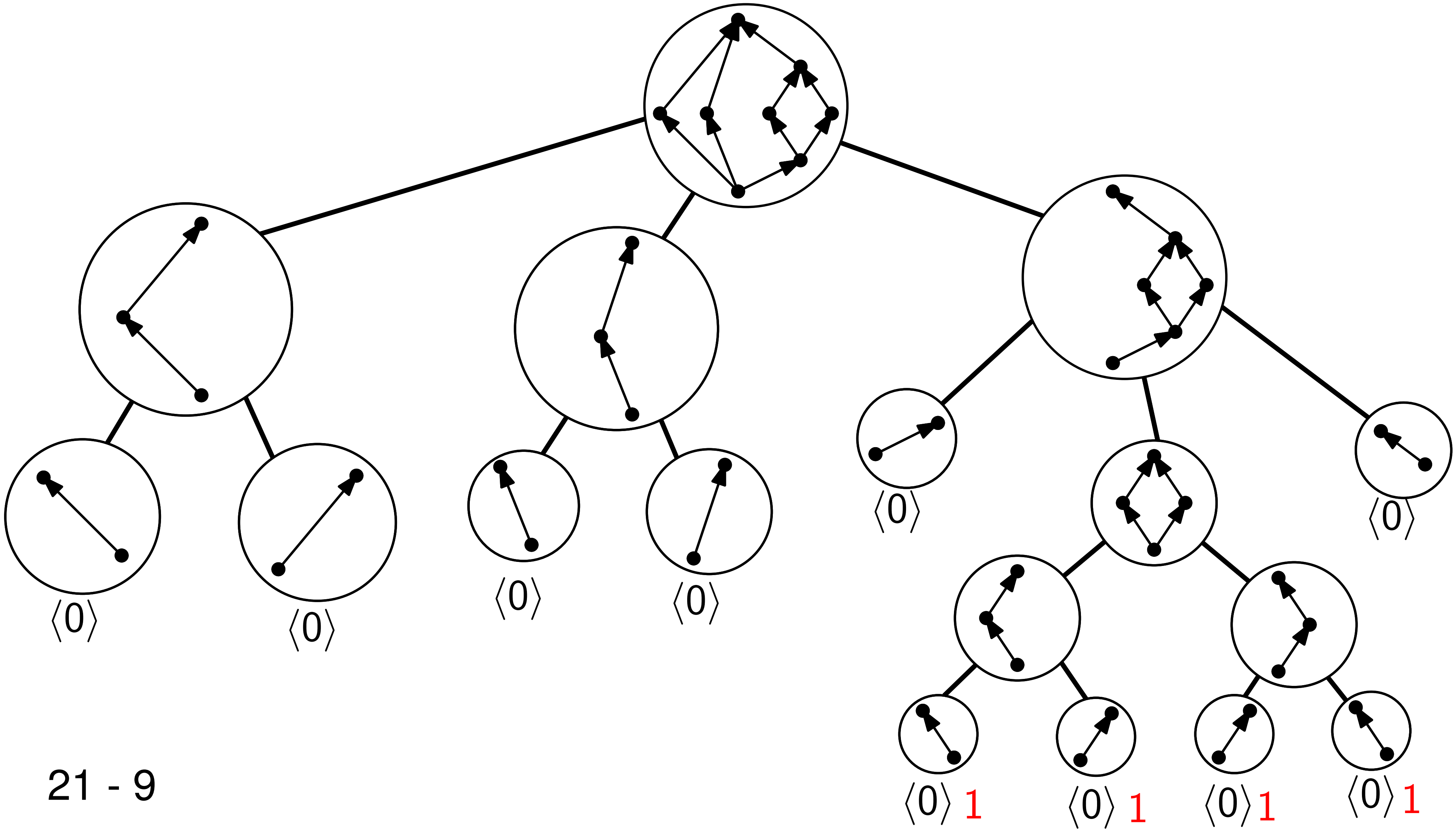
21 - 7

Vertical Automorphism



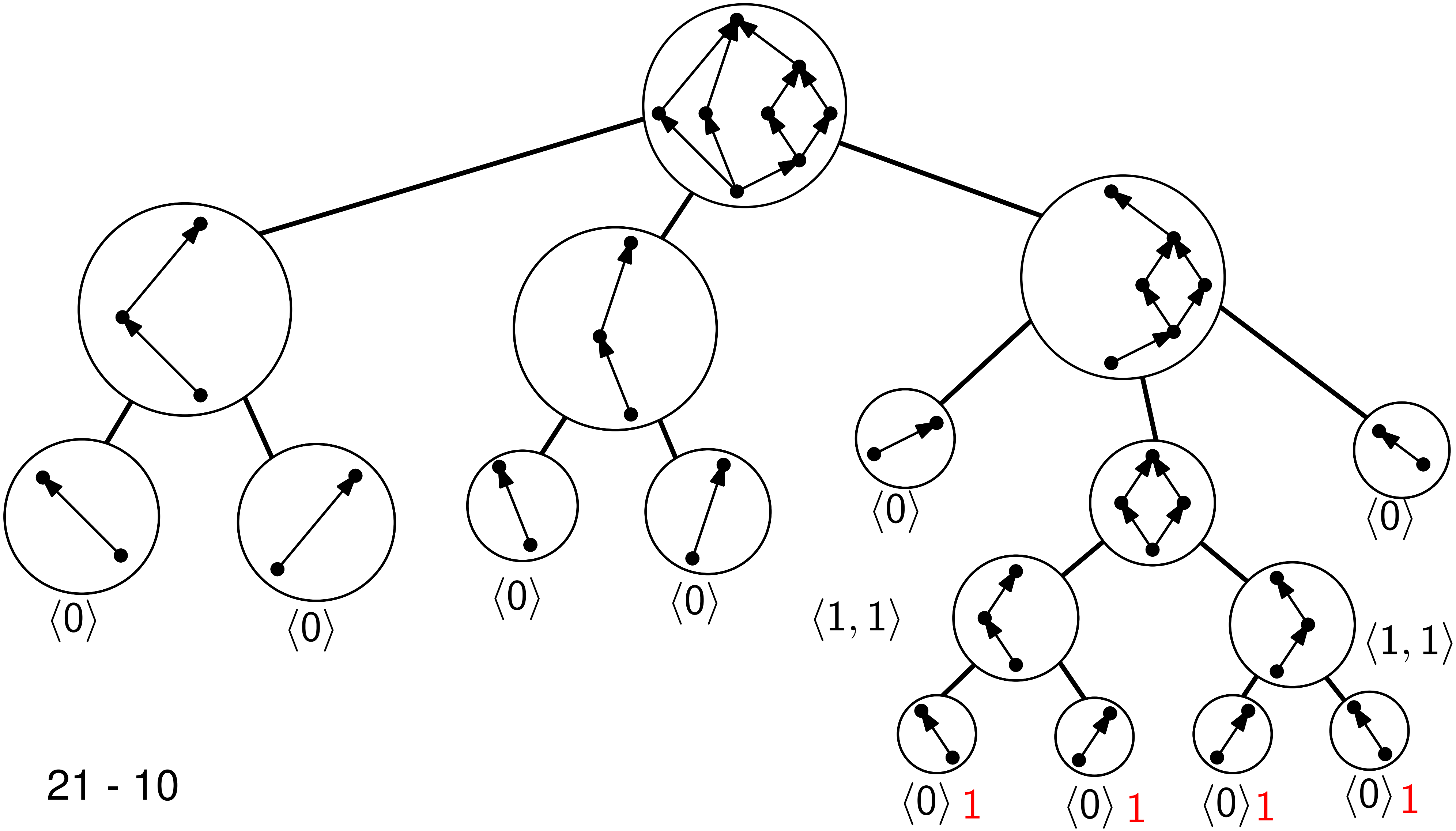
21 - 8

Vertical Automorphism



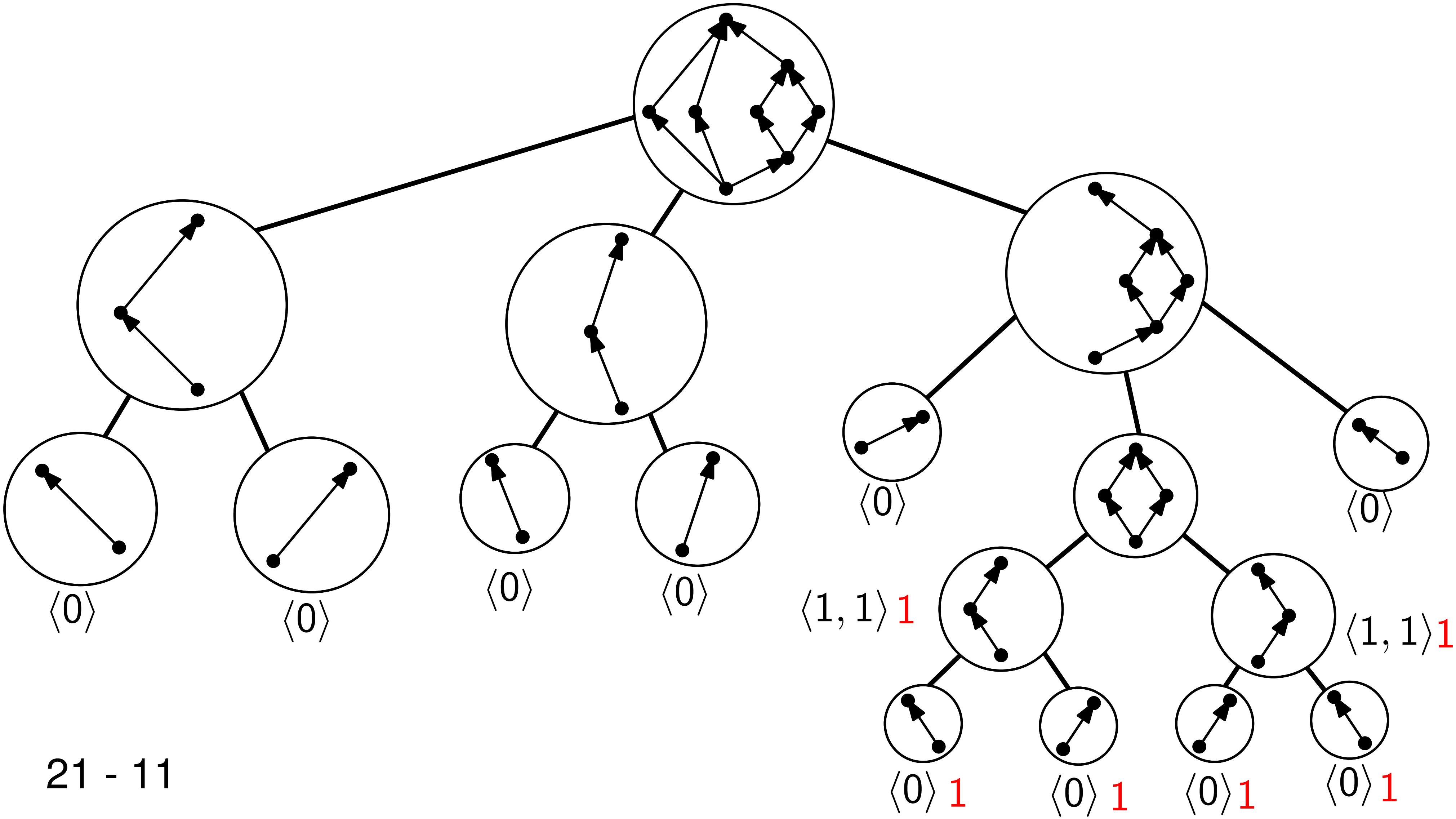
21 - 9

Vertical Automorphism



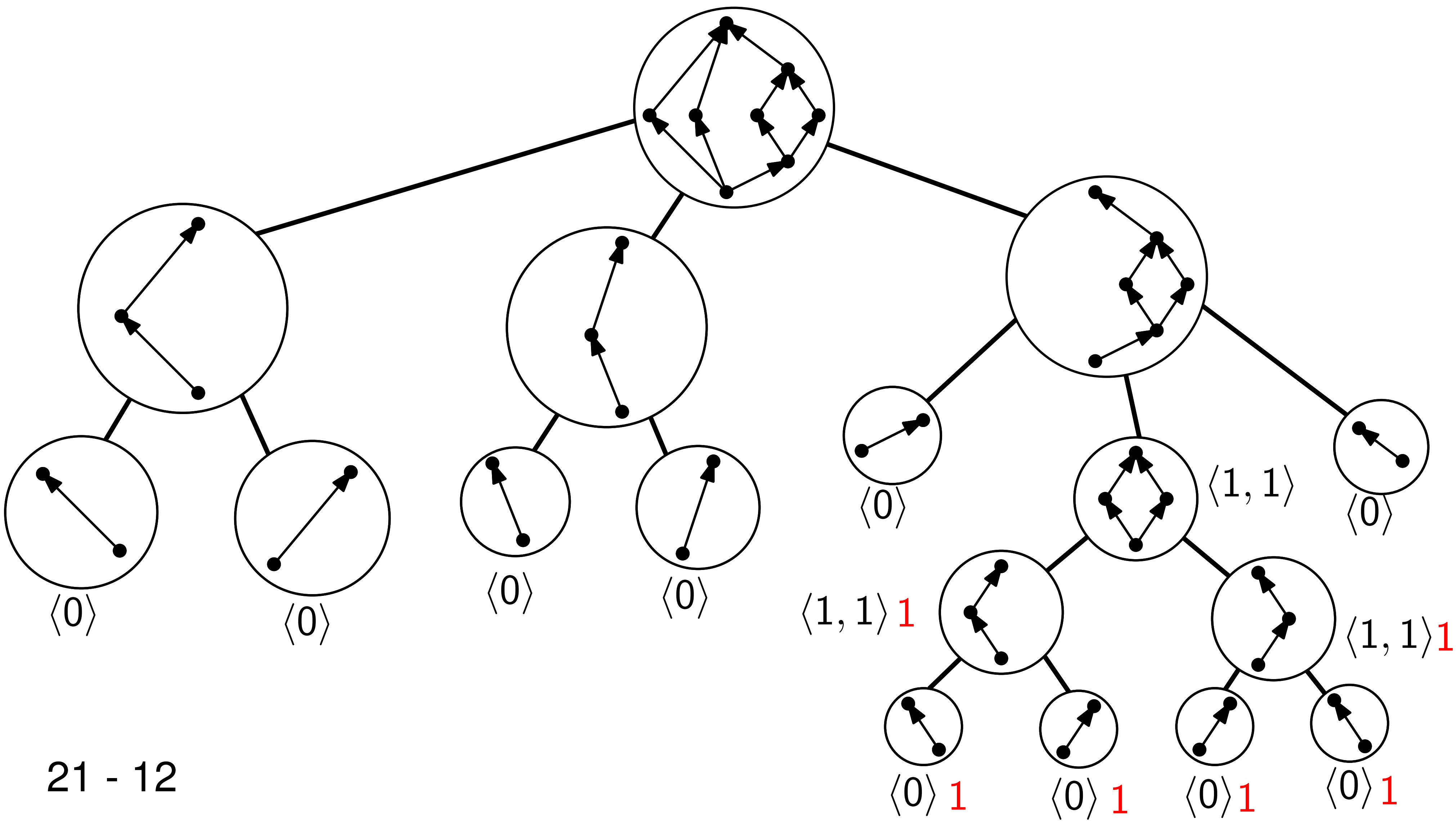
21 - 10

Vertical Automorphism

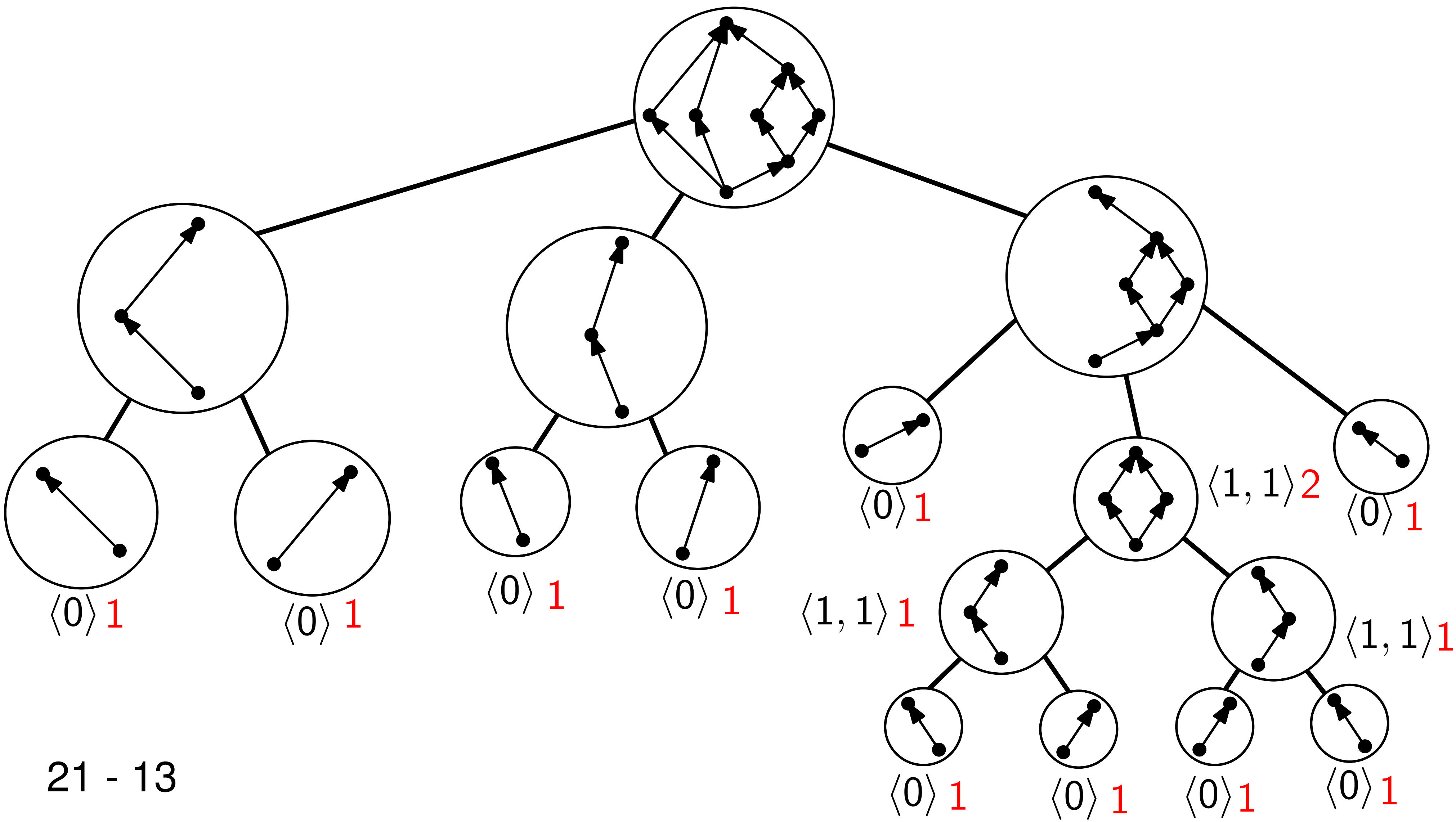


21 - 11

Vertical Automorphism

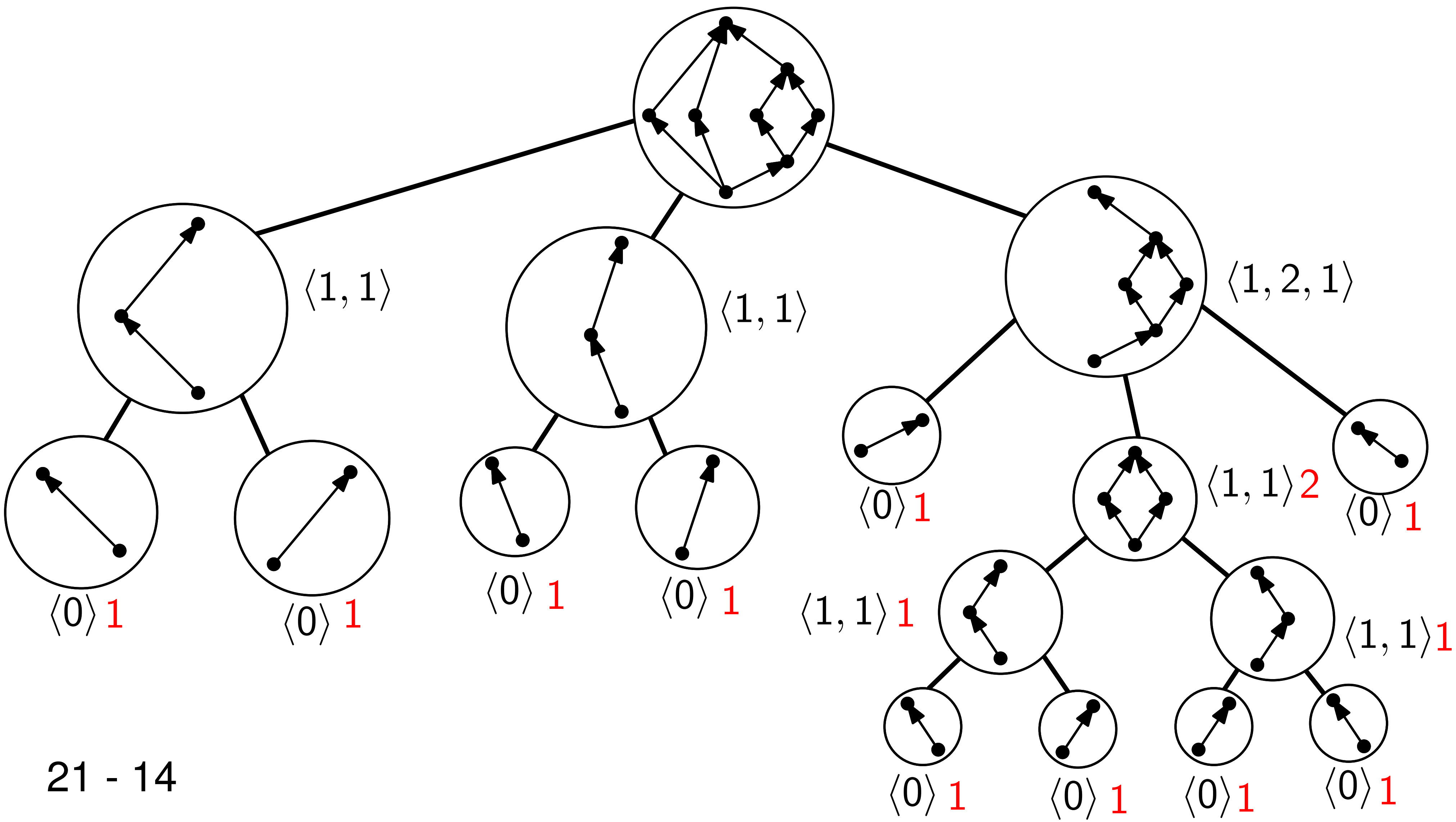


Vertical Automorphism

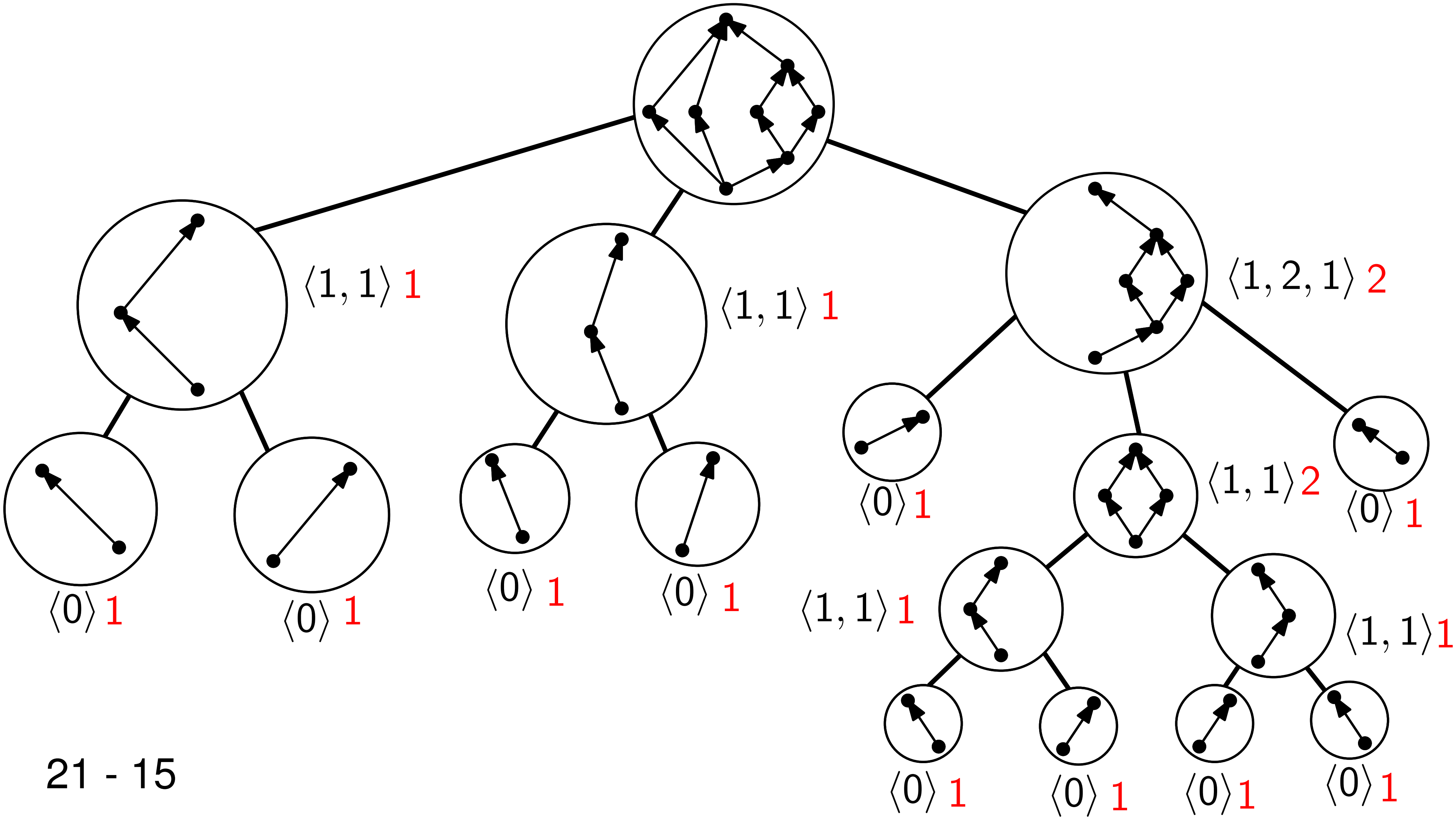


21 - 13

Vertical Automorphism

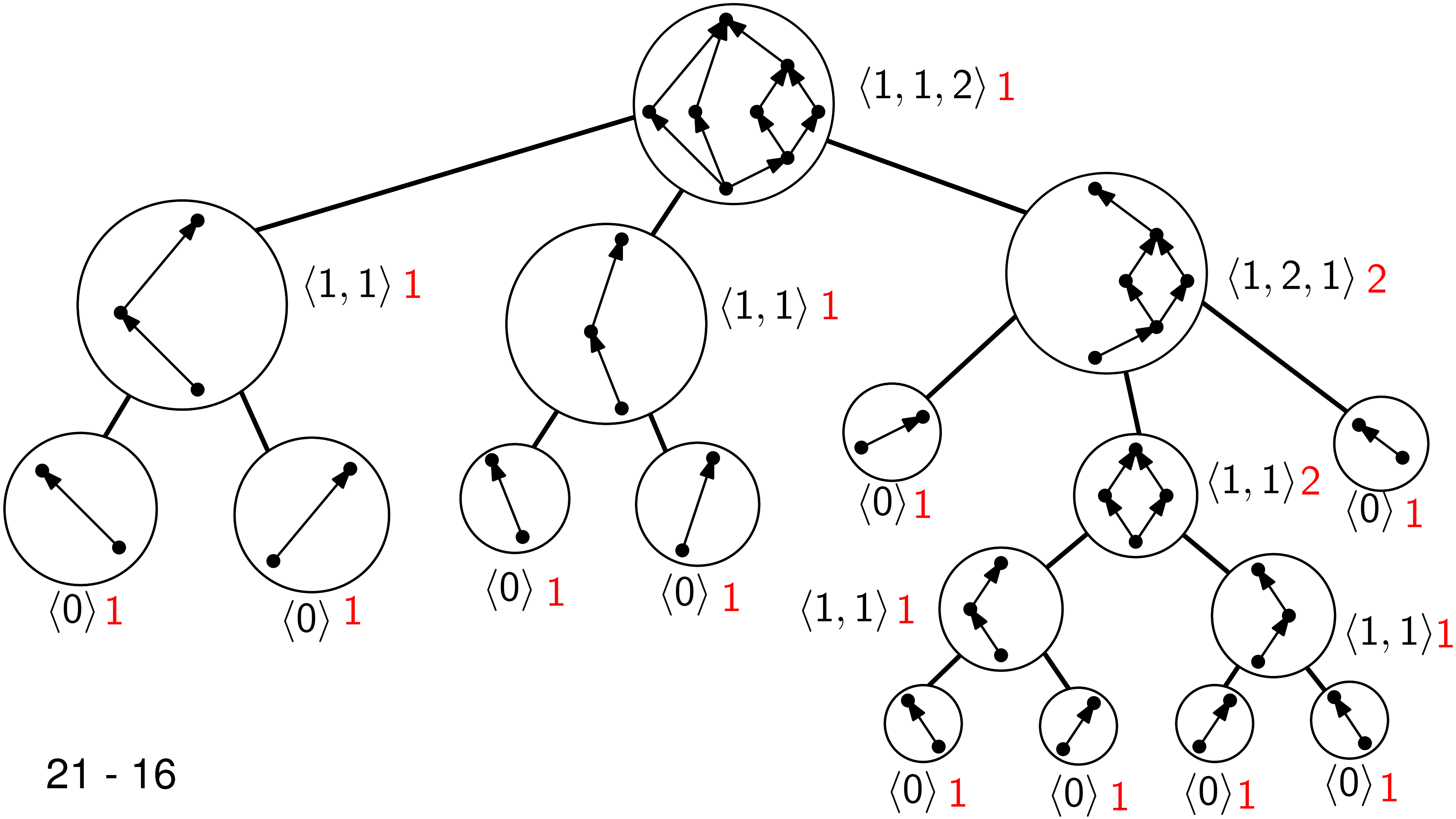


Vertical Automorphism



21 - 15

Vertical Automorphism



Vertical Automorphism

Algorithm constructing codes

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

Vertical Automorphism

Algorithm constructing codes

- 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 codes

- 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

Algorithm constructing codes

- Set $tuple(G_i) = \langle 0 \rangle$ for all Q-nodes G_i of G.
- For each $t = \max \mathrm{depth}(G), \ldots, 0$
  - 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 codes

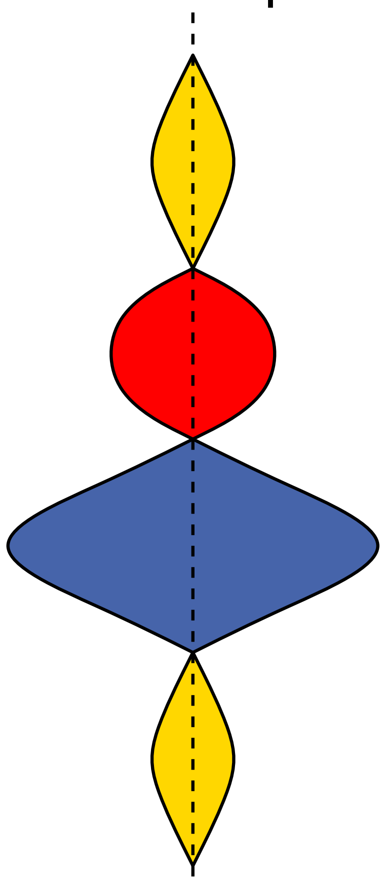
- Set $tuple(G_i) = \langle 0 \rangle$ for all Q-nodes G_i of G.
- For each $t = \max \text{depth}(G), \ldots, 0$
  - For each S- or P-node G' at depth t with children $G_1, \ldots, G_k$ set $tuple(G') = \langle code(G_1), \ldots, code(G_k) \rangle$. If G' is a P-node, sort $tuple(G')$ in non-decreasing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

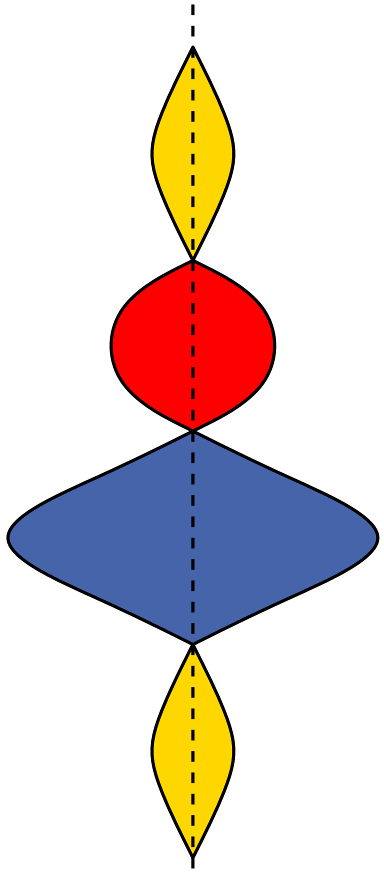
- Let G be composed out of $G_1 \ldots G_n$ through series or parallel composition, $tuple(G)$ contains the codes of $G_1, \ldots, G_n$.
- How can we use $tuple(G)$ do decide whether G has a vertical automorphism?



23 G 1 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?



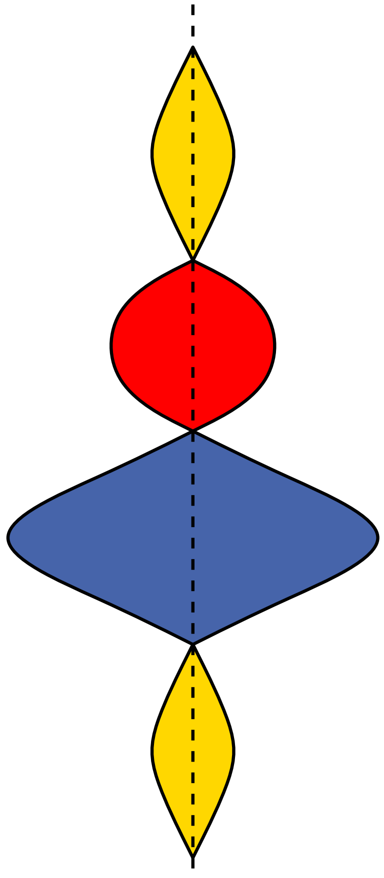
G is an S-node

Lemma (Hong, Eades, Lee ’00) [HEL00]

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) [HEL00]

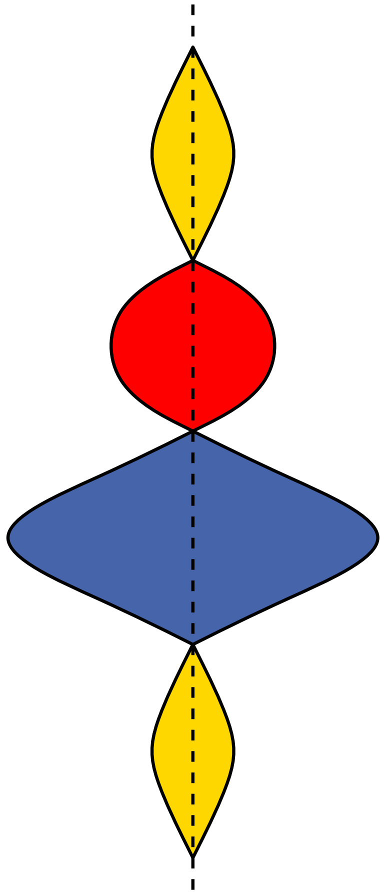
If G is an S-node, then G has a vertical automorphism iff each of $G_1, \ldots, G_k$ has a vertical automorphism.

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?



23 G 4s an S-node

Lemma (Hong, Eades, Lee ’00) [HEL00]

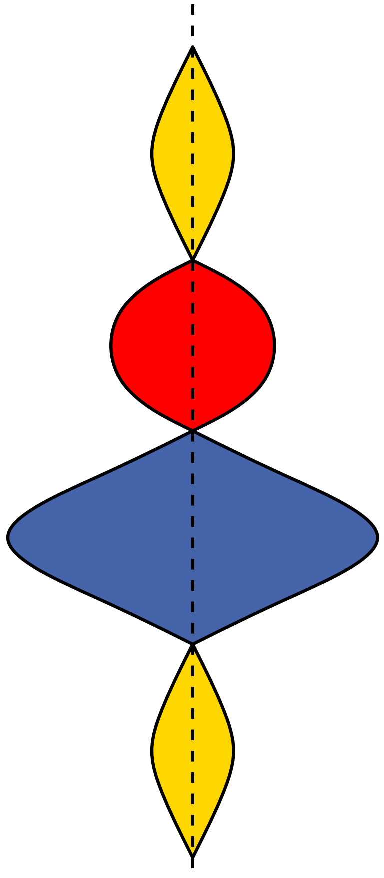
If G is an S-node, then G has a vertical automorphism iff each of $G_1, \dots, G_k$ has a vertical automorphism.

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) [HEL00]

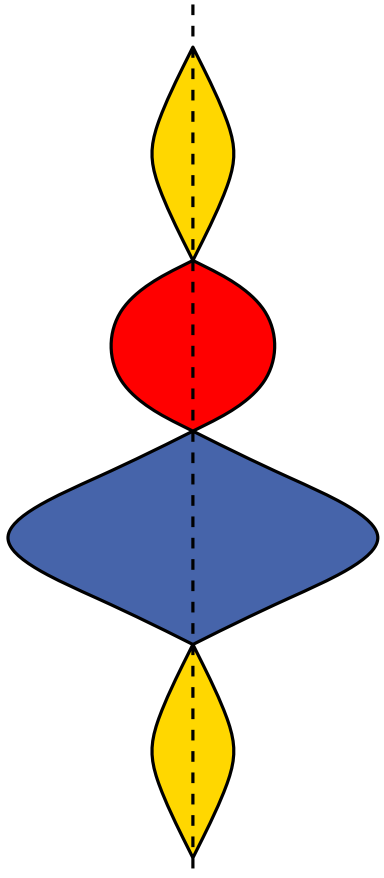
If G is an S-node, then G has a vertical automorphism iff each of $G_1, \dots, G_k$ has a vertical automorphism.

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 $G_1 \dots G_n$ 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) [HEL00]

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:

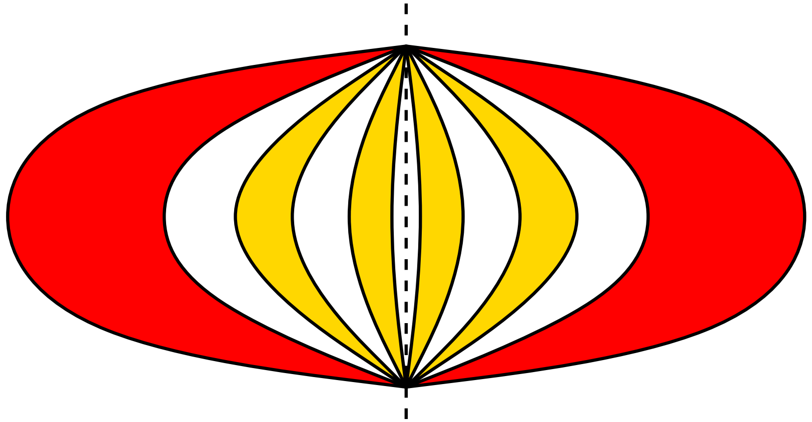
- Assume G has a vertical automorphism α
- Then α "fixes" all the components
- Therefore each of the series components has a vertical automorhism
- If each of $G_1, \dots, G_n$ has a vertical isomorphism, arrange them as in Figure.

Vertical Automorphism

Lemma (Hong, Eades, Lee ’00) [HEL00]

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.

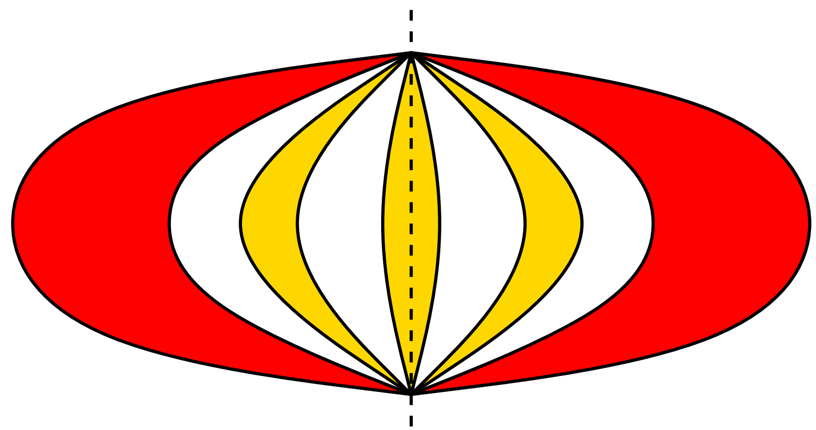
G is P-node, $tuple(G) = < \underbrace{1 \ldots 1}_{even}, \underbrace{2 \ldots 2}_{even}, \cdots >$

Vertical Automorphism

Lemma (Hong, Eades, Lee '00) [HEL00]

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:

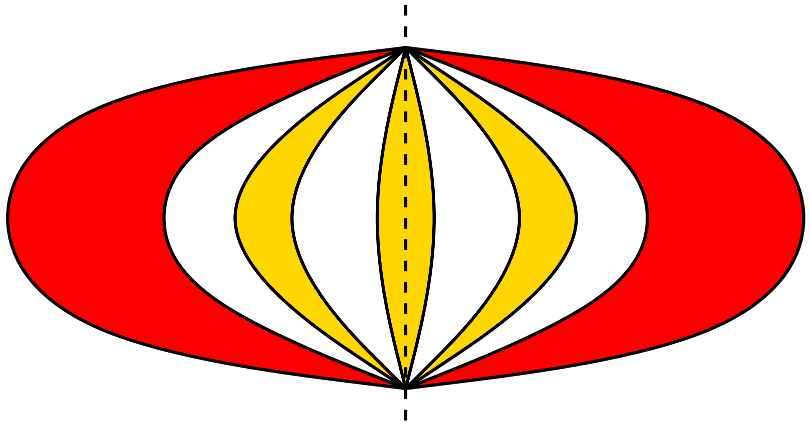
$$tuple(G) = < \underbrace{1 \ldots 1}_{odd}, \underbrace{2 \ldots 2}_{even}, \underbrace{3 \ldots 3}_{even}, \cdots >$$

Vertical Automorphism

Lemma (Hong, Eades, Lee ’00) [HEL00]

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.



$$tuple(G) =< \underbrace{1\ldots1}_{odd}, \underbrace{2\ldots2}_{even}, \underbrace{3\ldots3}_{even}, \cdots >$$

Proof:

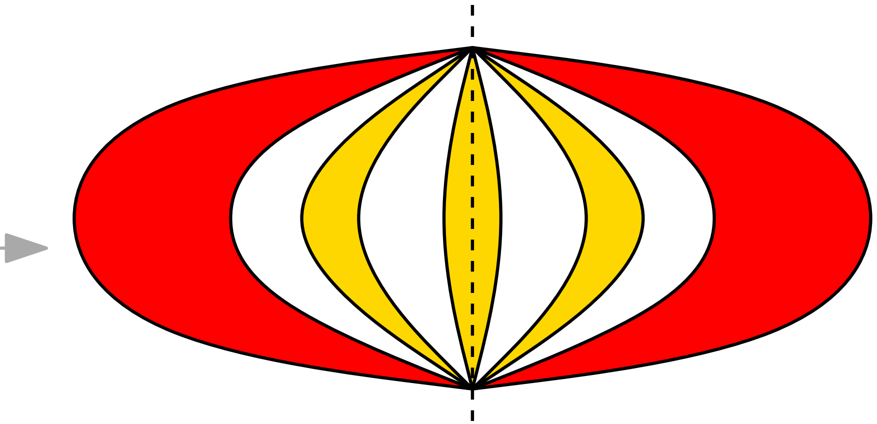
- Any vertical automorphism “fixes” a member of $\mathcal{C}_j$, therefore it has a vertical automorphism.

Vertical Automorphism

Lemma (Hong, Eades, Lee '00) [HEL00]

If G is a P-node, consider a partition of $\mathcal{C}_j = \{G_i : 1 \le i \le k, code(G_i) = j\}$, $j = 1, \ldots, k$ 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 $i \ne j$, both odd $\Rightarrow$ G does not have a vertical automorphism.



$$tuple(G) =< \underbrace{1\ldots1}_{odd}, \underbrace{2\ldots2}_{even}, \underbrace{3\ldots3}_{even}, \cdots >$$

Proof:

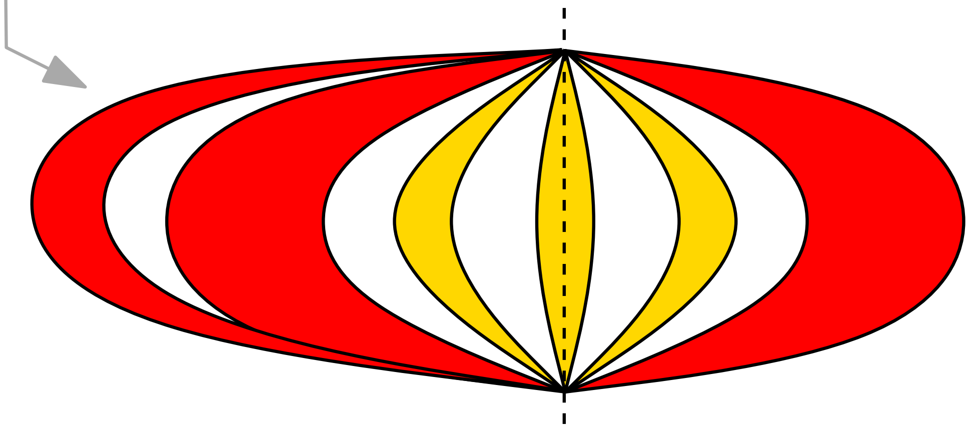
- Any vertical automorphism "fixes" a member of $\mathcal{C}_j$, therefore it has a vertical automorphism.
- Conversly, arrange as in figure.

Vertical Automorphism

Lemma (Hong, Eades, Lee '00) [HEL00]

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.



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

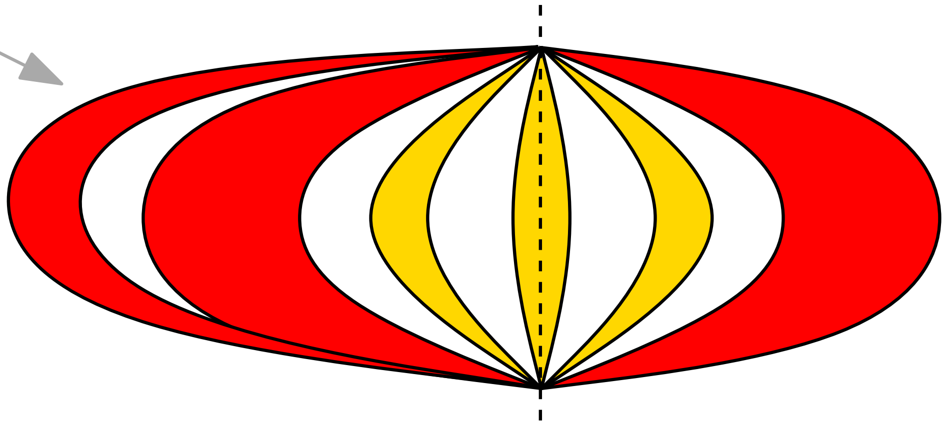
Proof:

Vertical Automorphism

Lemma (Hong, Eades, Lee '00) [HEL00]

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:

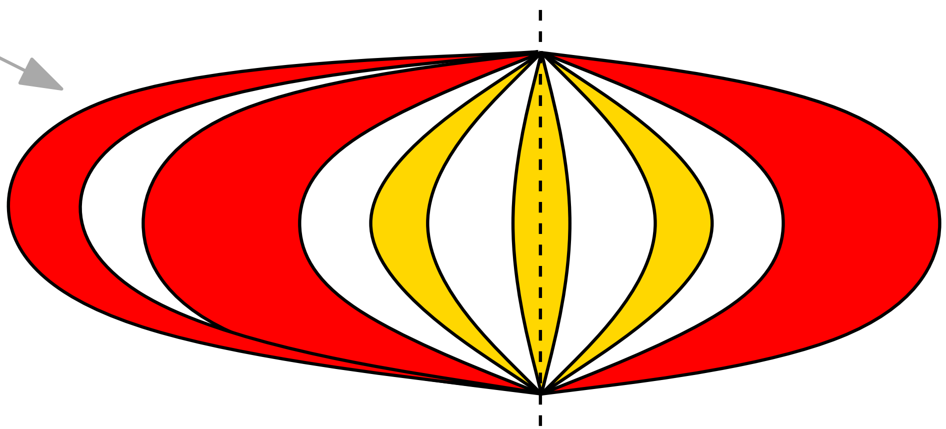
- Any vertical automorphism has to "fix" two distinct components.

Vertical Automorphism

Lemma (Hong, Eades, Lee '00) [HEL00]

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.

Series-parallel graphs

- Book Di Battista et al: Chapter 3.2, 11.1
- Skript: Chapter 6.2
- [HEL00] Hong, Eades, Lee **Drawing series parallel digraphs symetrically** CGTA 2000