Algorithmen zur Visualisierung von Graphen

Teile & Herrsche-Algorithmen: Bäume und serien-parallele Graphen

Vorlesung im Wintersemester 2012/2013

Tamara Mchedlidze – Martin Nöllenburg – Ignaz Rutter

23. Oktober 2012

Institute for Theoretical Informatics
Algorithmics Group I

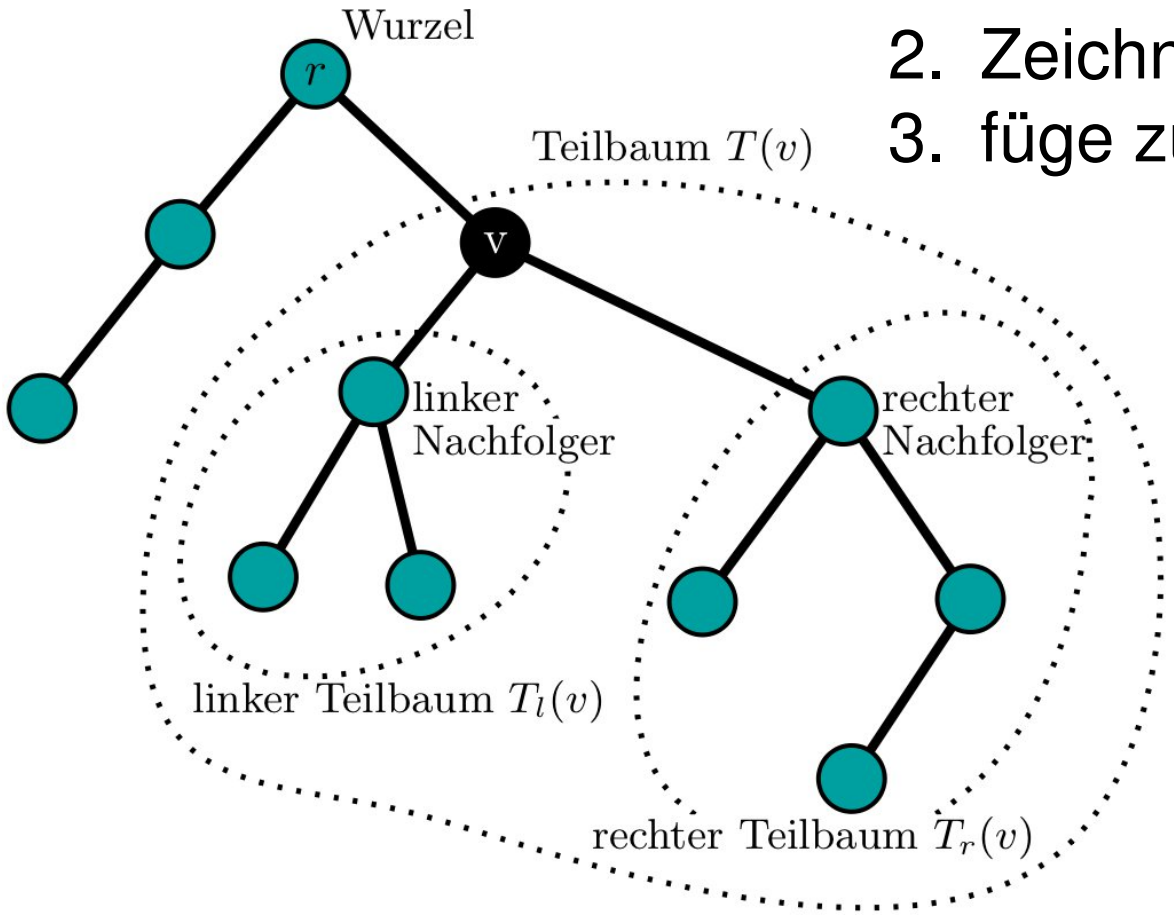
Algorithmen zum Zeichnen von Bäumen

Institute for Theoretical Informatics
Algorithmics Group I

Anwendbarkeit

Gut bei induktiv oder rekursiv definierten Familien von Graphen

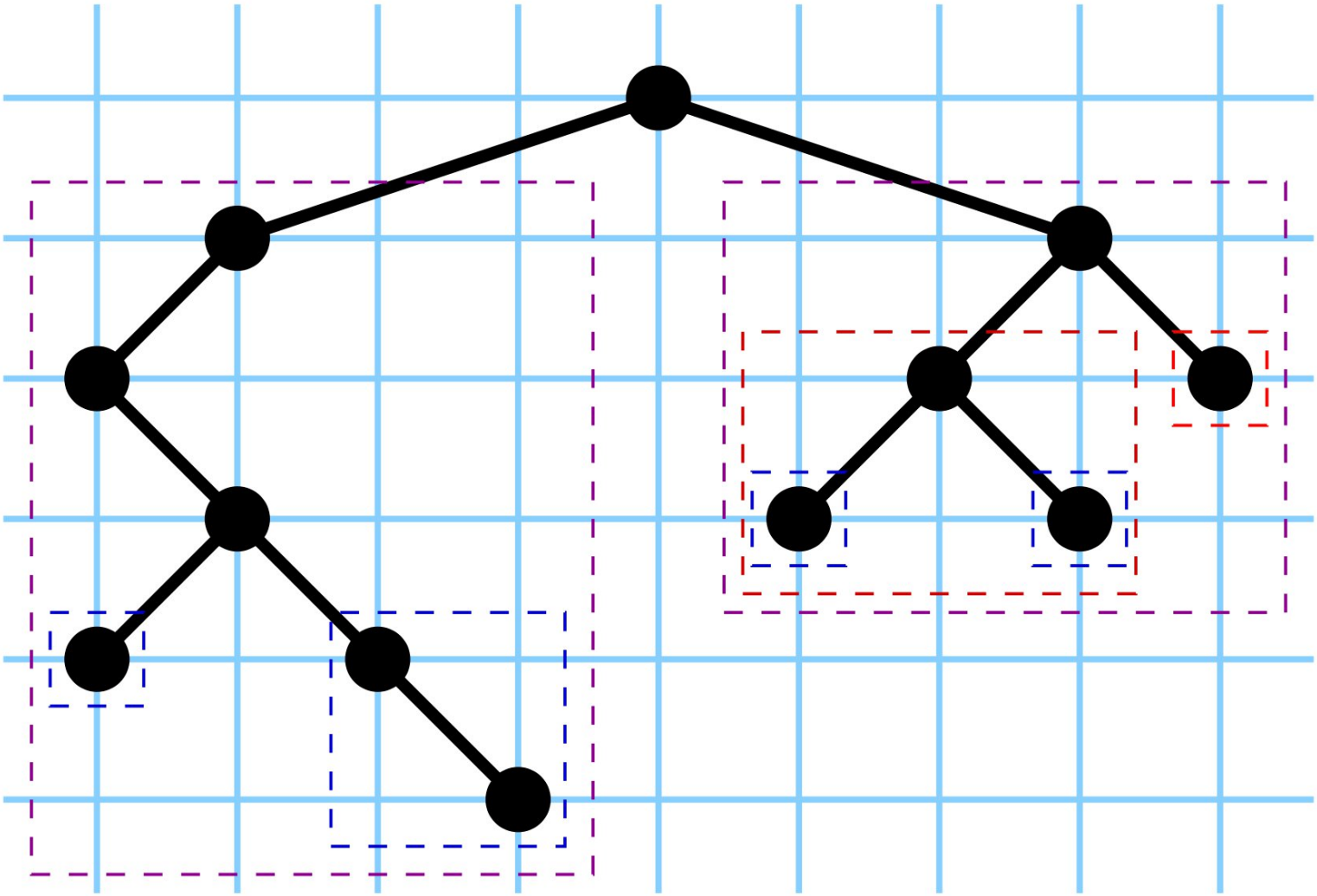
Binärbaum mit Wurzel:



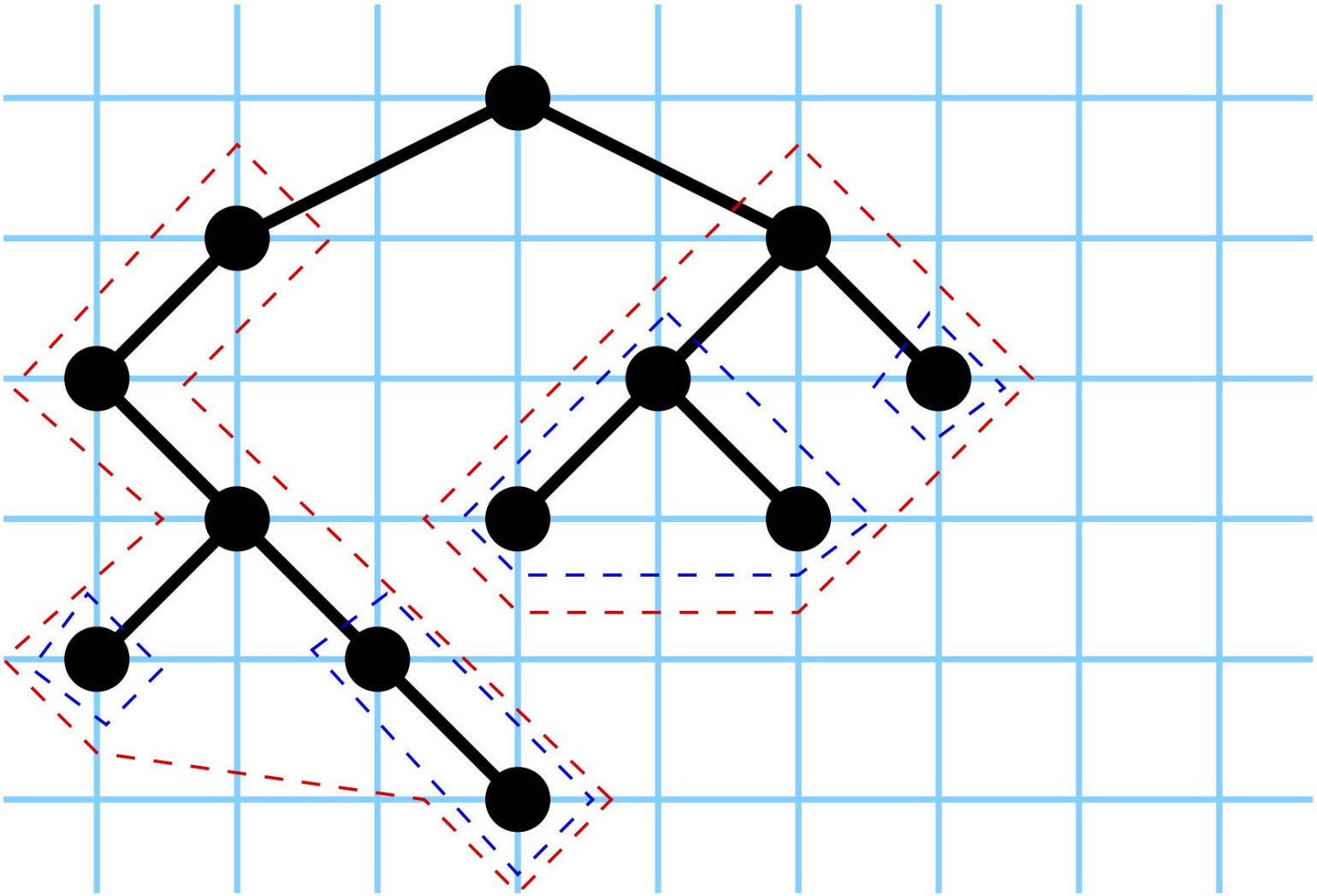
1. Zeichne linken Teilbaum
2. Zeichne rechten Teilbaum
3. füge zusammen + Wurzel

- tiefe(v): Abstand zur Wurzel
- Durchlauf-reihenfolgen
  - preorder
  - inorder
  - postorder

Algorithmus von Reingold und Tilford (’81)



Algorithmus von Reingold und Tilford (’81)

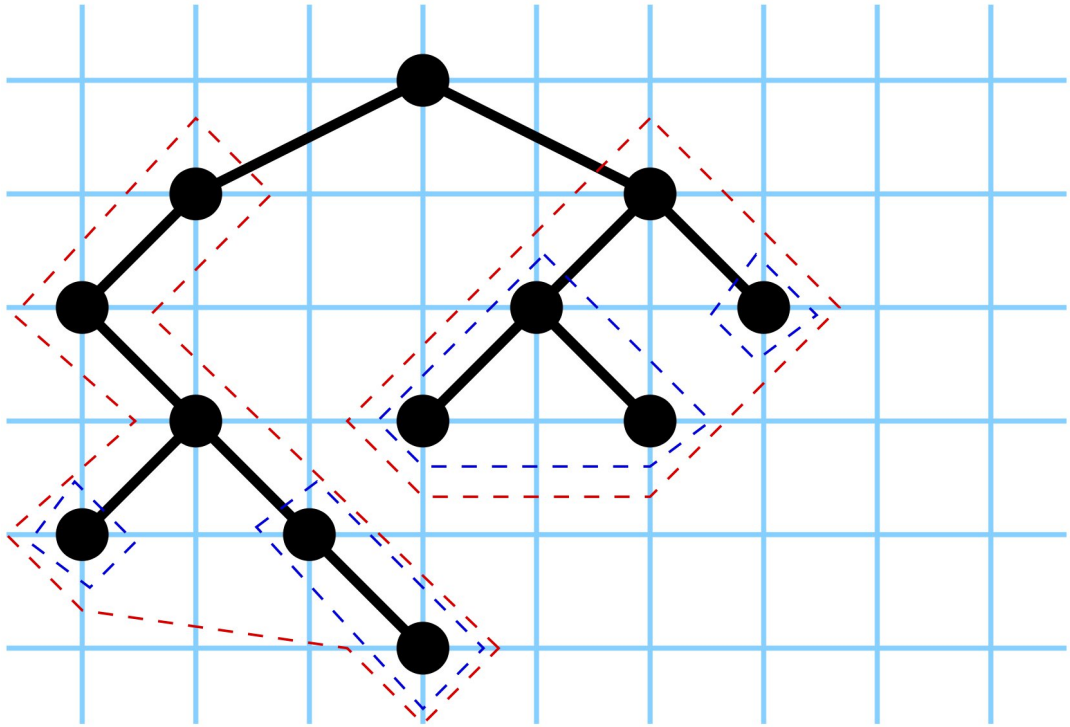


Institute for Theoretical Informatics
Algorithmics Group I

Algorithmus von Reingold und Tilford (’81)

2 Phasen:

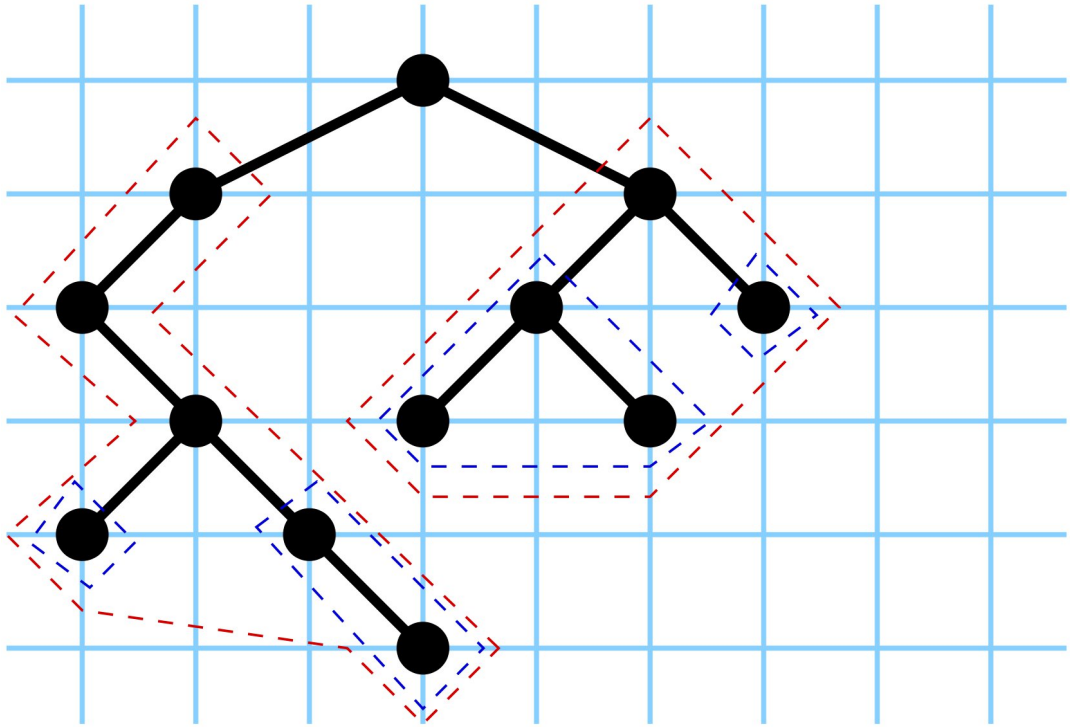
1. postorder (bottom-up): Konturen und x-Offsets zum Vorgänger einsammeln
2. preorder (top-down): absolute Koordinaten ausrechnen



Algorithmus von Reingold und Tilford ('81)

2 Phasen:

1. postorder (bottom-up): Konturen und x-Offsets zum Vorgänger einsammeln
2. preorder (top-down): absolute Koordinaten ausrechnen



Kontur: verkettete Liste von Knoten (-Koordinaten)

Algorithmus von Reingold und Tilford ('81)

Phase 1:

1. Bearbeite $T_\ell(v)$ und $T_r(v)$
2. Laufe parallel linke Kontur von $T_r(v)$ und rechte Kontur von $T_\ell(v)$ ab
3. Bestimmt daraus d_v, den horizontalen Minimalabstand von v_ℓ und v_r
4. x-Offset(v_ℓ) = $-\lceil \frac{d_v}{2} \rceil$, x-Offset($v_r$) = $\lceil \frac{d_v}{2} \rceil$
5. Baue linke Kontur von T_v aus: v, linke Kontur von $T_\ell(v)$ und evtl. überhängendes Teilstück von linker Kontur von $T_r(v)$
6. Rechte Kontur analog

Algorithmus von Reingold und Tilford ('81)

Phase 2

1. Setze y-Koordinate $y(v) = -\text{tiefe}(v)$
2. Setze $x(v) = 0$ für Wurzel und rekursiv die x-Koordinate $x(v_\ell)$ und $x(v_r)$ der Nachfolger von v auf $x(v)+ x\text{-Offset}(x(v_\ell))$ bzw. $x(v)+ x\text{-Offset}(x(v_r))$

Zusammenfassung:
Algorithmus berechnet Binärbaumlayout:

- geradliniges Gitterlayout
- tiefengeschichtet, kreuzungsfrei
- Knoten derselben Tiefe haben Abstand ≥ 2
- Knoten sind über Nachfolgern zentriert
- linke/rechte Nachfolger sind strikt links/rechts
- identische Teilbäume gleich gezeichnet

Breitenminimierung von Binärbaumlayouts

Satz (Supowit, Reingold)

Die Breitenminimierung von Binärbaumlayouts ist NP-schwer

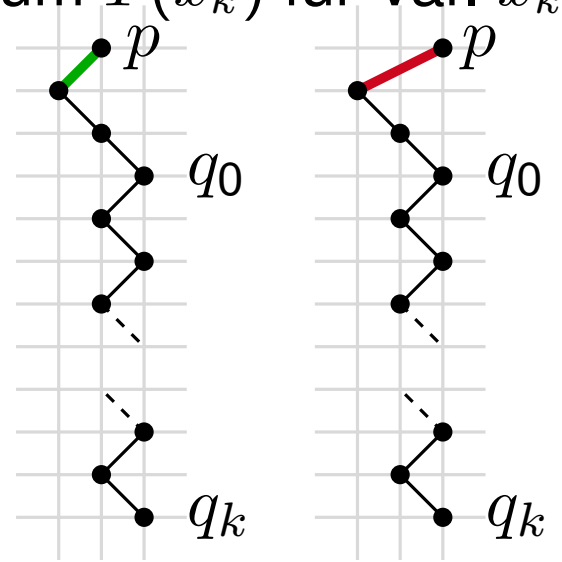
Beweis: Reduktion von 3SAT

$F = C_1 \wedge \cdots \wedge C_m$,

$C_i = y_{i,1} \vee y_{i,2} \vee y_{i,3}, \quad y_{i,j} \in \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$

Konstruiere Baum $T(F)$, der genau dann Layout mit Breite $W \leq 24$ hat, wenn F erfüllbar ist.

Baum $T(x_k)$ für Var. x_k :



Breitenminimierung von Binärbaumlayouts

Satz (Supowit, Reingold)

Die Breitenminimierung von Binärbaumlayouts ist NP-schwer

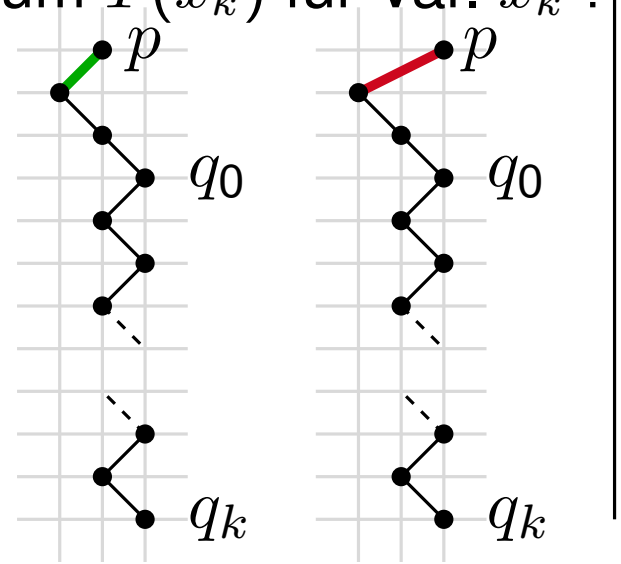
Beweis: Reduktion von 3SAT

$F = C_1 \wedge \cdots \wedge C_m$,

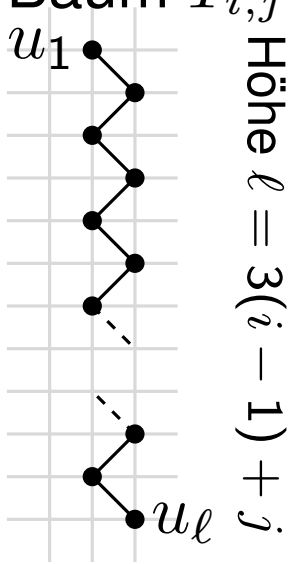
$C_i = y_{i,1} \vee y_{i,2} \vee y_{i,3}, \quad y_{i,j} \in \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$

Konstruiere Baum $T(F)$, der genau dann Layout mit Breite $W \leq 24$ hat, wenn F erfüllbar ist.

Baum $T(x_k)$ für Var. x_k :



Baum $T_{i,j}$



Höhe $\ell = 3(i-1) + j$.

Breitenminimierung von Binärbaumlayouts

Satz (Supowit, Reingold)

Die Breitenminimierung von Binärbaumlayouts ist NP-schwer

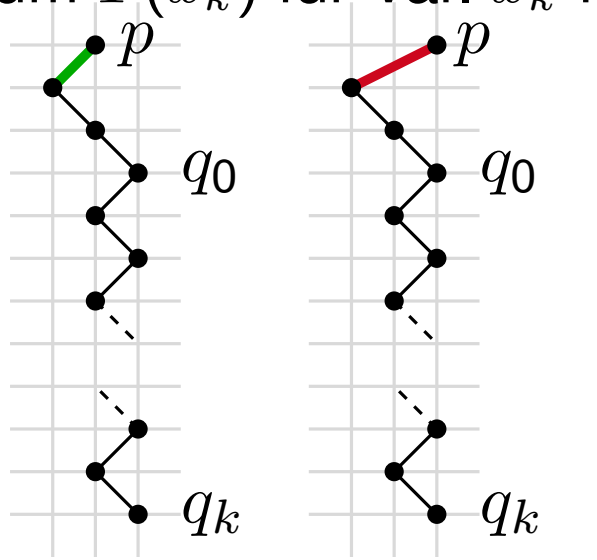
Beweis: Reduktion von 3SAT

$F = C_1 \wedge \cdots \wedge C_m$,

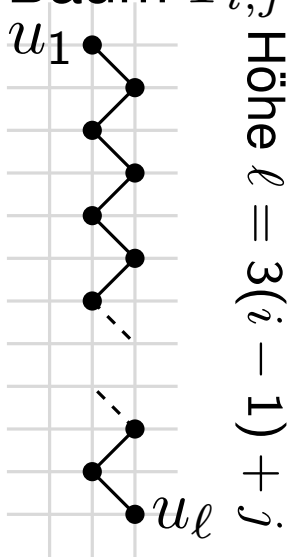
$C_i = y_{i,1} \vee y_{i,2} \vee y_{i,3}, \quad y_{i,j} \in \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$

Konstruiere Baum $T(F)$, der genau dann Layout mit Breite $W \leq 24$ hat, wenn F erfüllbar ist.

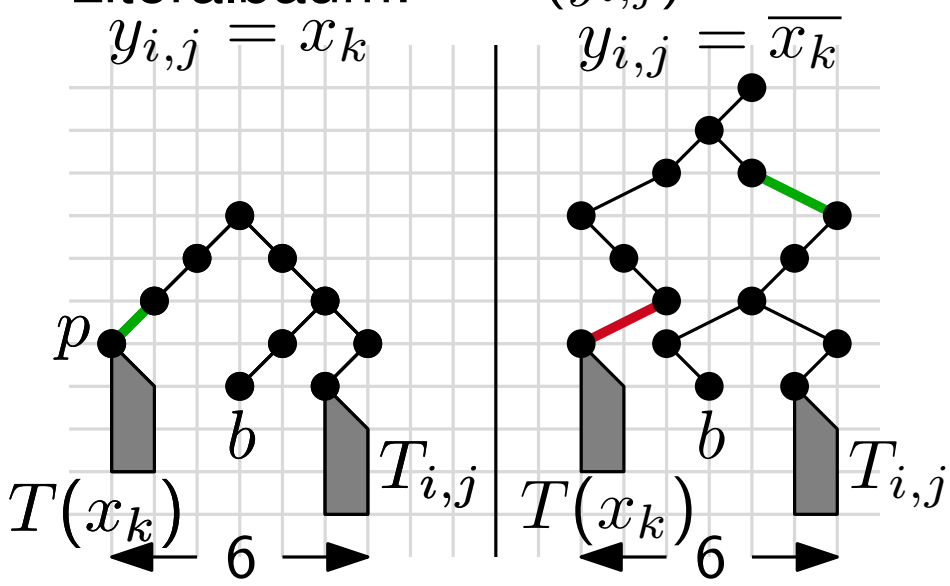
Baum $T(x_k)$ für Var. x_k :



Baum $T_{i,j}$



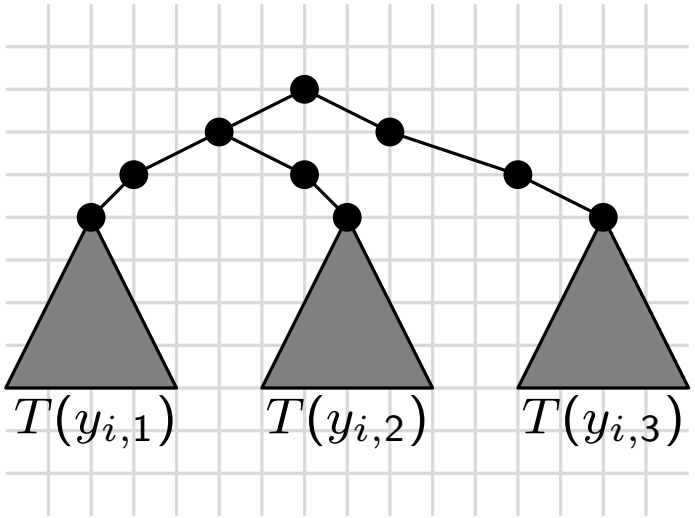
Literalbaum:



$T(y_{i,j})$ für

Breitenminierung II

Klauselbaum $T(C_i)$:



Erfüllte Klausel hat Breite höchstens

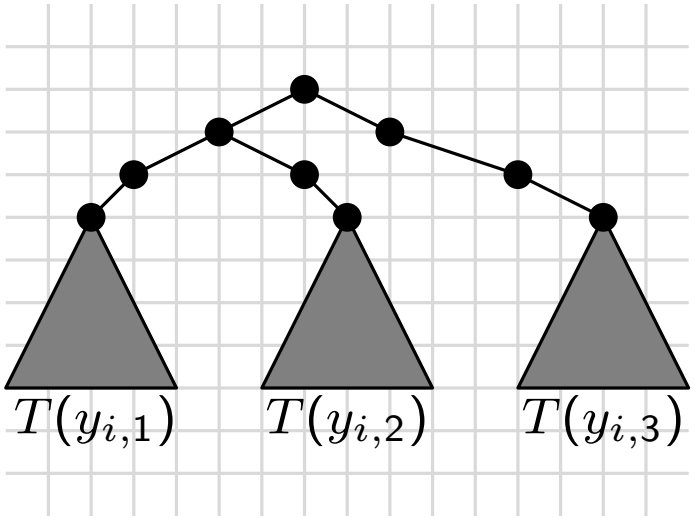
$$6 + 2 + 7 + 2 + 7 = 24$$

Beachte: alle Literalbäume haben volle Breite auf vierter Ebene von oben

Nicht erfüllte Klausel: 7+2+7+2+7 = 25

Breitenminierung II

Klauselbaum $T(C_i)$:



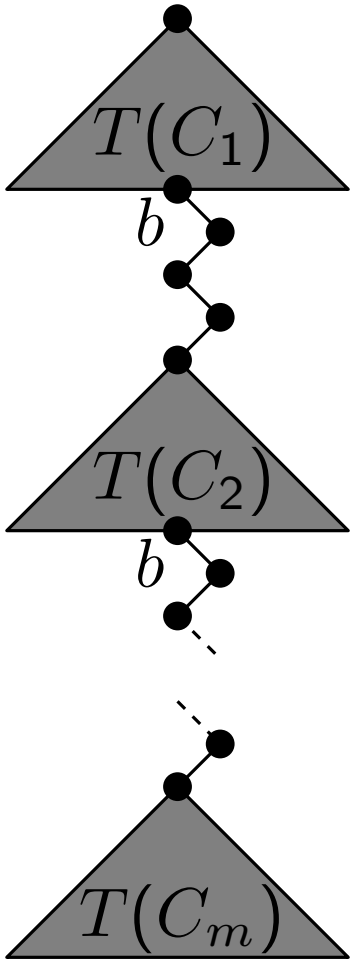
Erfüllte Klausel hat Breite höchstens

$$6 + 2 + 7 + 2 + 7 = 24$$

Beachte: alle Literalbäume haben volle Breite auf vierter Ebene von oben

Nicht erfüllte Klausel: 7+2+7+2+7 = 25

Formelbaum $T(F)$



Breite $\leq 24 \Leftrightarrow F$ erfüllbar

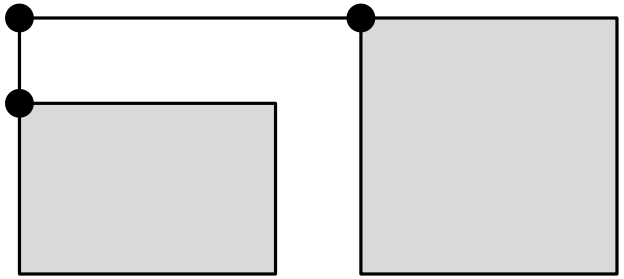
HV-Bäume

Idee:

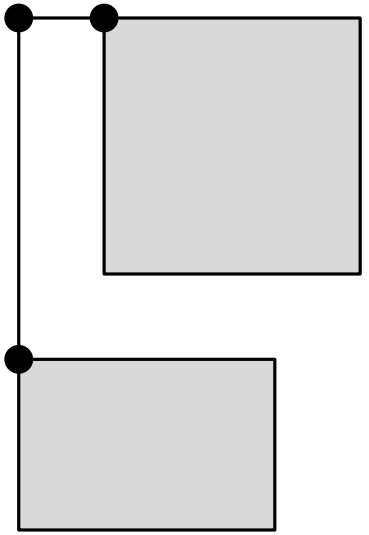
- Zeichne Teilbäume in Rechtecke, Wurzel liegt in linker oberer Ecke
- Nachfolger liegen vertikal unterhalb bzw. horizontal rechts

Induktionsanfang:

Induktionsschritt: kombiniere Layouts



horizontale Kombination
(Fläche: 3×7)



vertikale Kombination
(Fläche: 4×6)

Berechne optimale Zeichnung mit dynamischer Programmierung

Rechtslastige hv-Layouts

Rechtslastiges hv-Layout:

- Wähle in jedem Schritt Horizontal-Kombination
- Platziere größeren Teilbaum rechts

Lemma

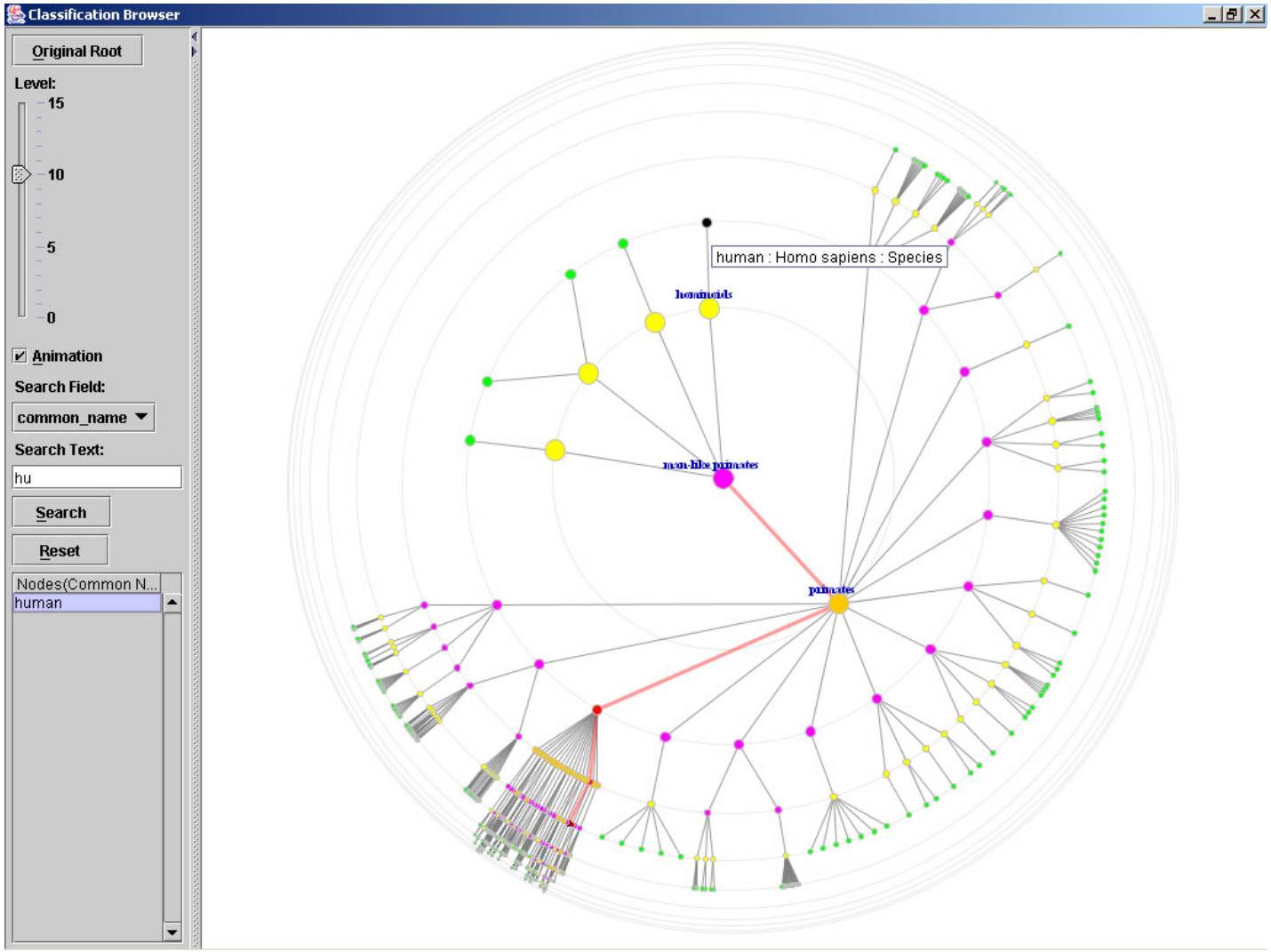
Höhe eines rechtslastigen hv-Layouts für Baum mit n Knoten ist höchstens $\log n$.

Beweis:

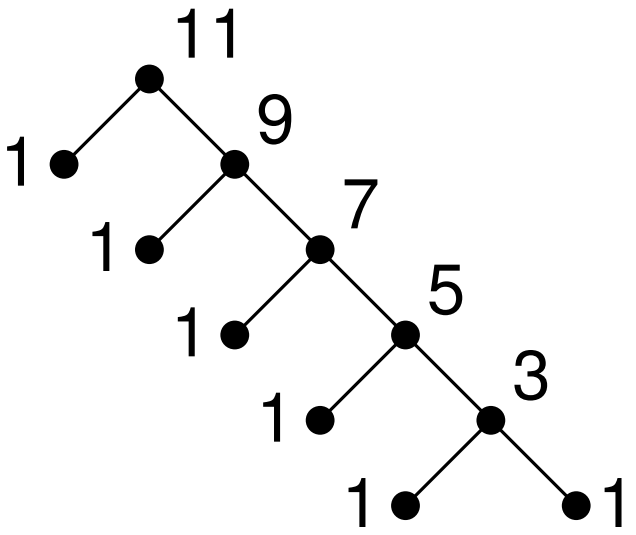
- Vertikale Kanten haben Länge 1
- w Knoten mit minimaler y-Koordinate
- betrachte eindeutigen Pfad P zur Wurzel
- für jede vertikale Kante (u, v) auf P: $|T(v)| > |2T(u)|$
- $\Rightarrow P$ enthält höchstens $\log n$ solcher Kanten

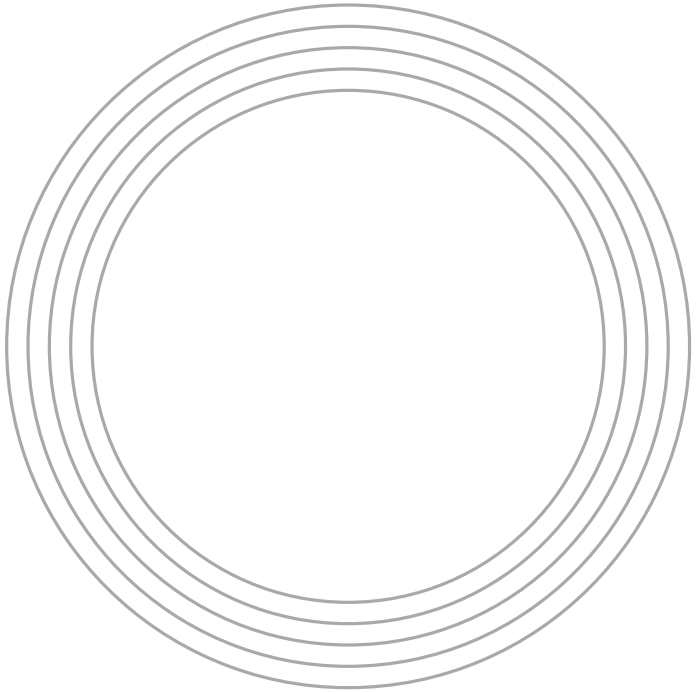
Platzbedarf: $O(n \log n)$

Radiale Baumlayouts

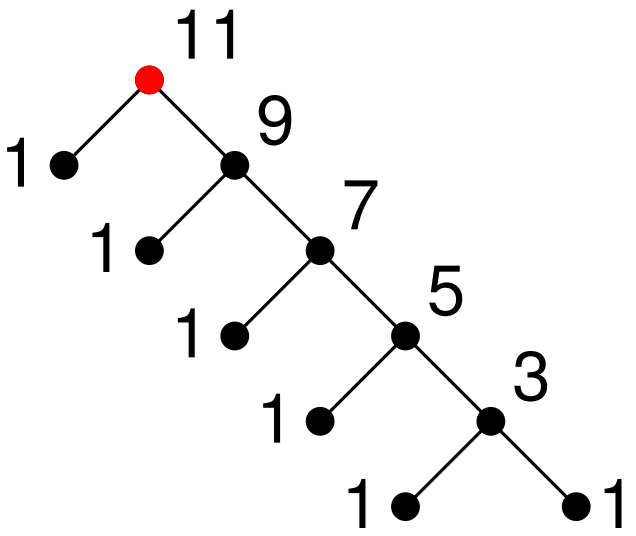


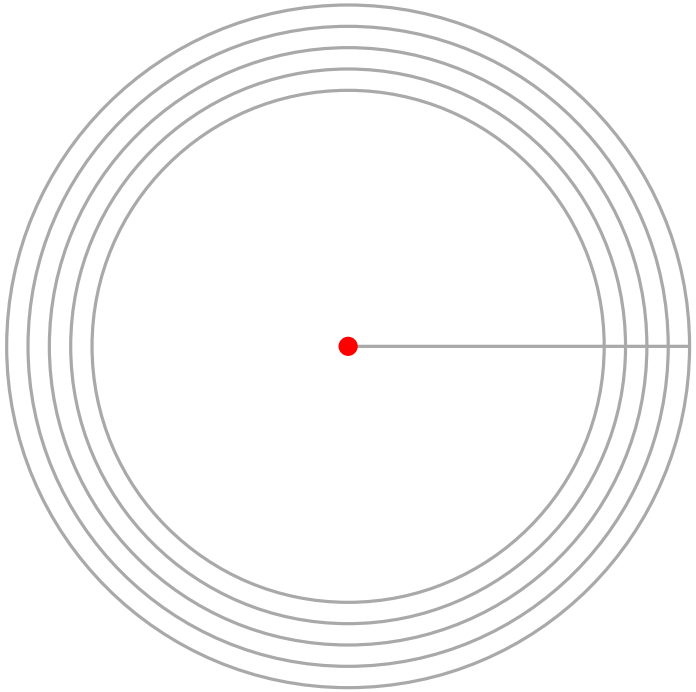
Beispiel Radiallayout



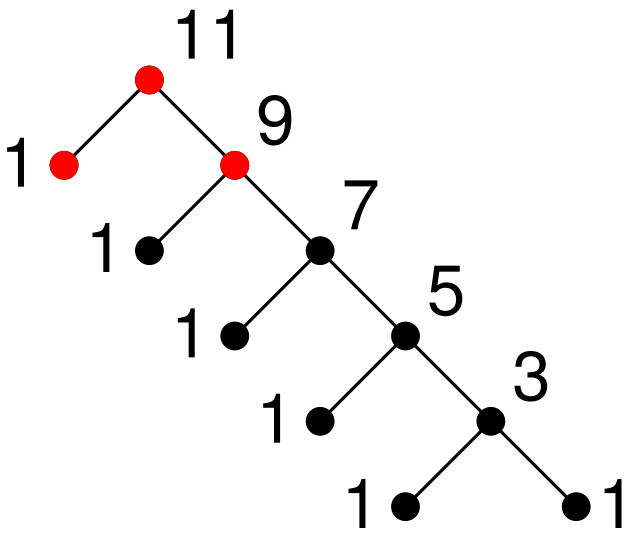


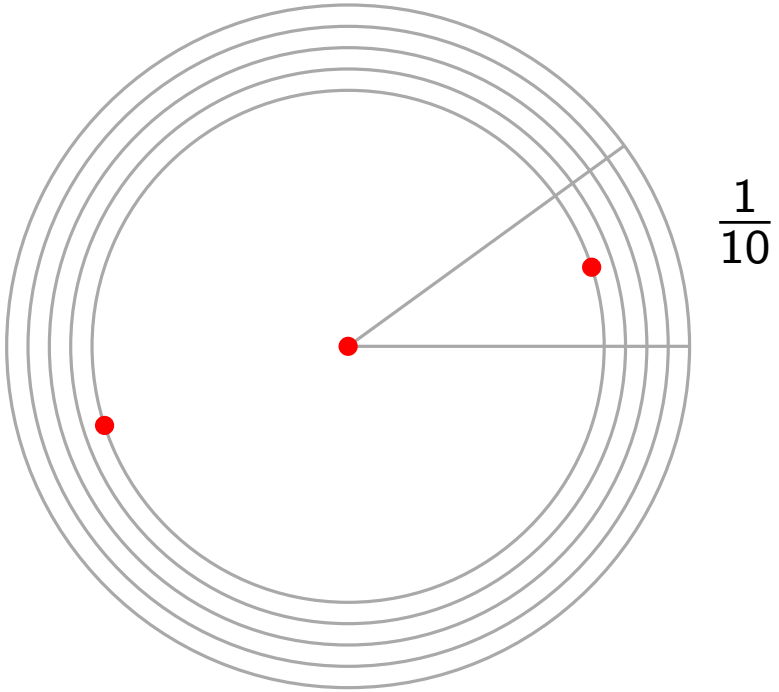
Beispiel Radiallayout



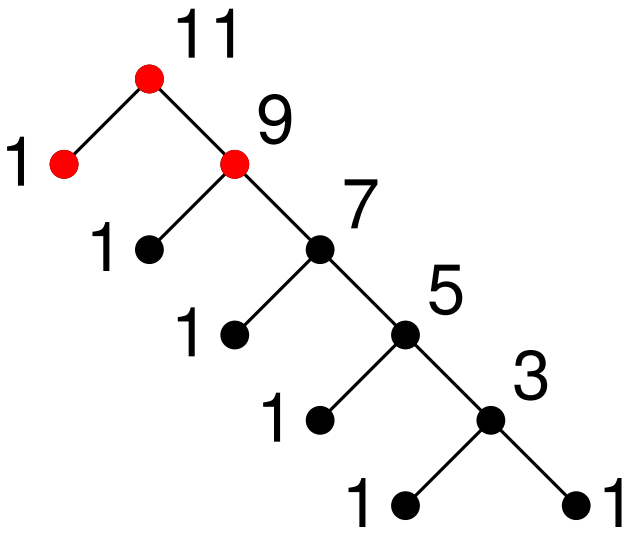


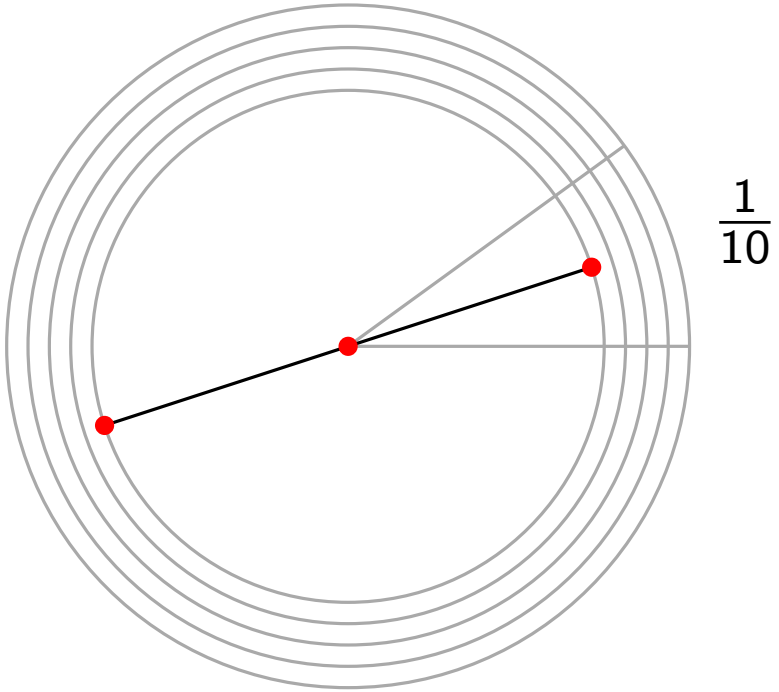
Beispiel Radiallayout



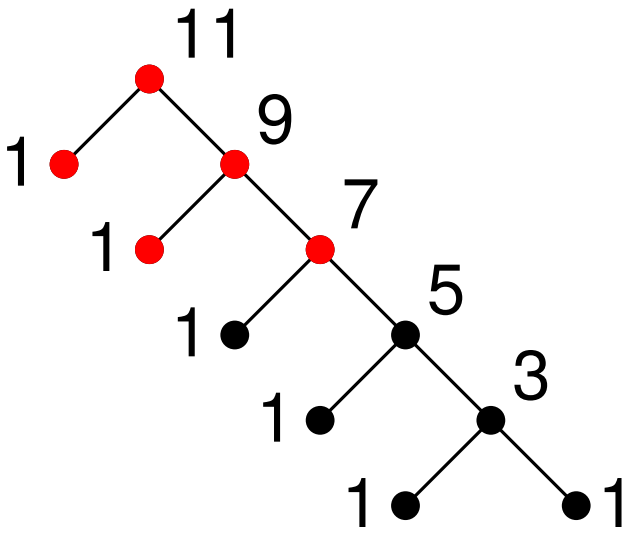


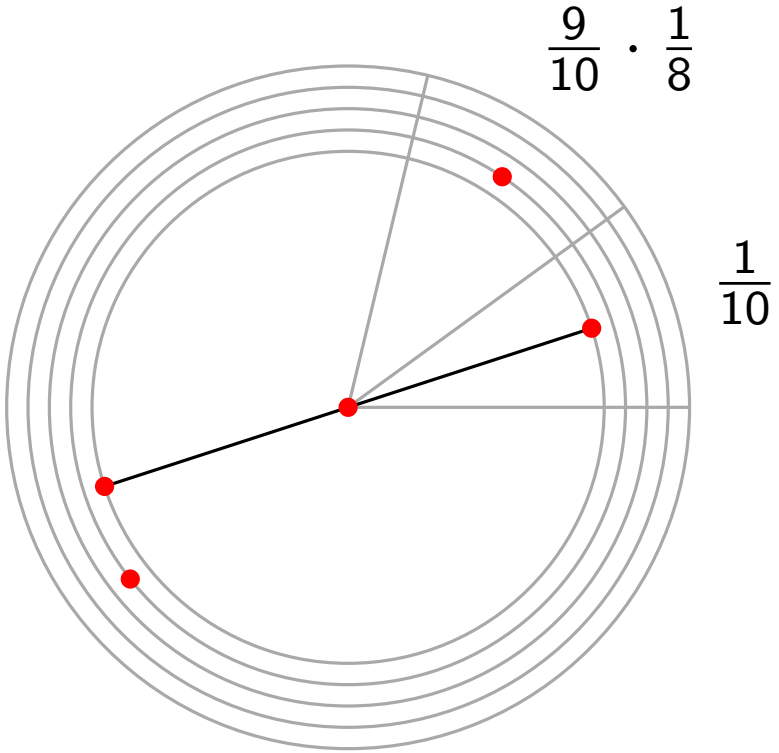
Beispiel Radiallayout



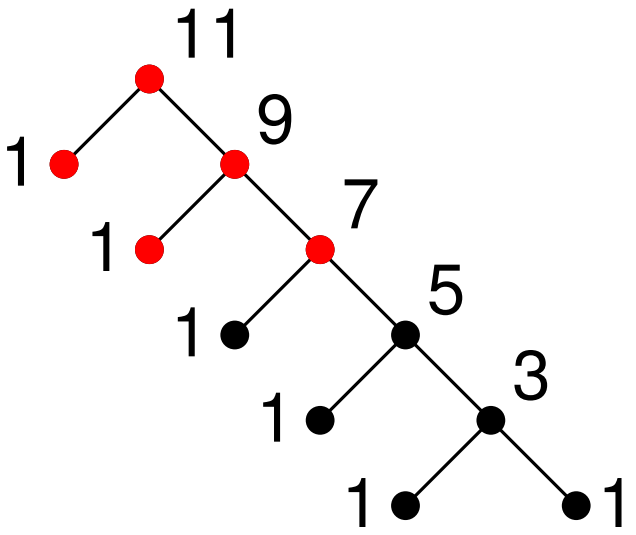


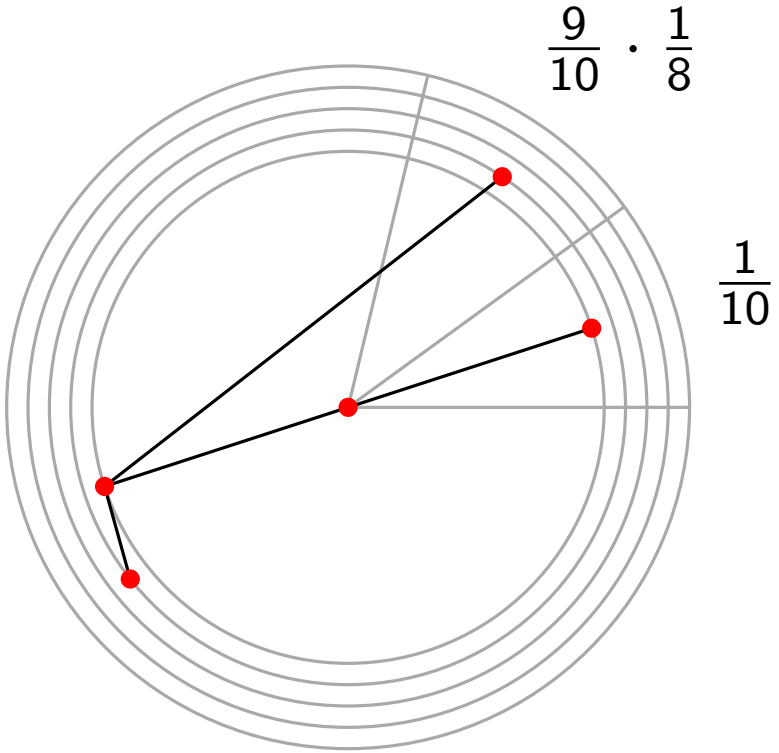
Beispiel Radiallayout



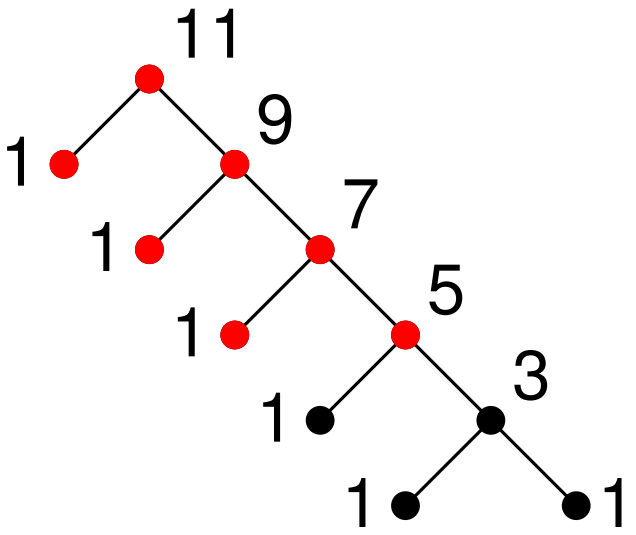


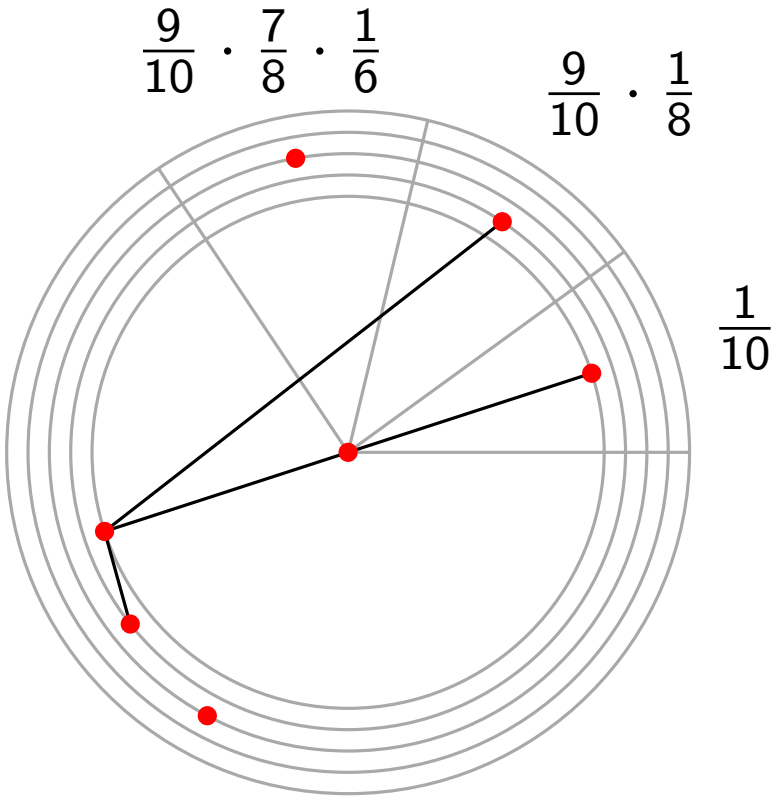
Beispiel Radiallayout



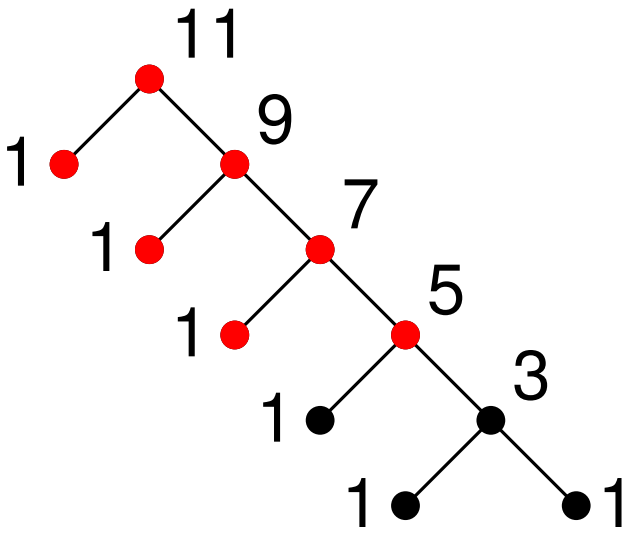


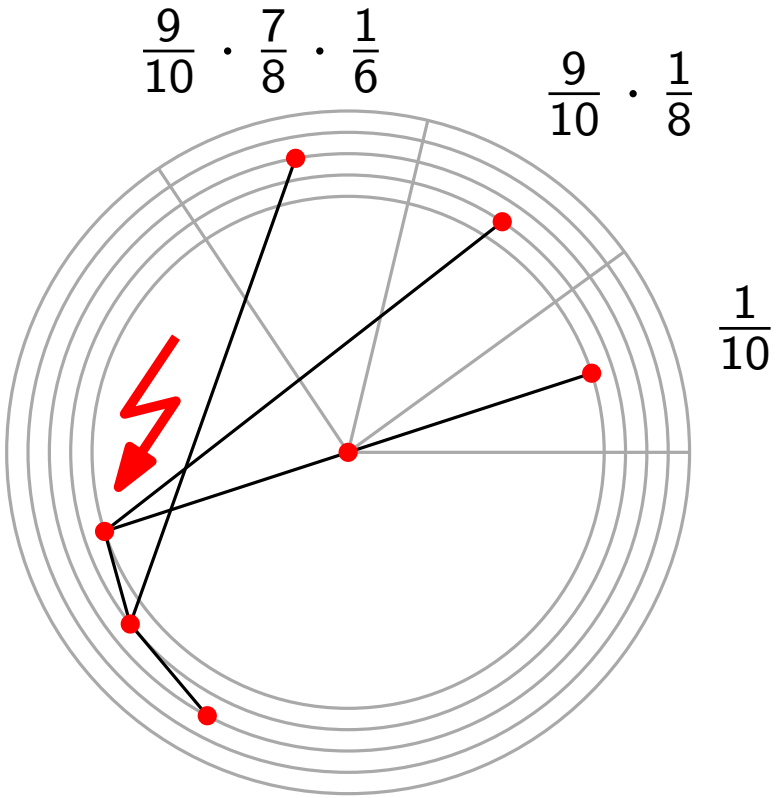
Beispiel Radiallayout



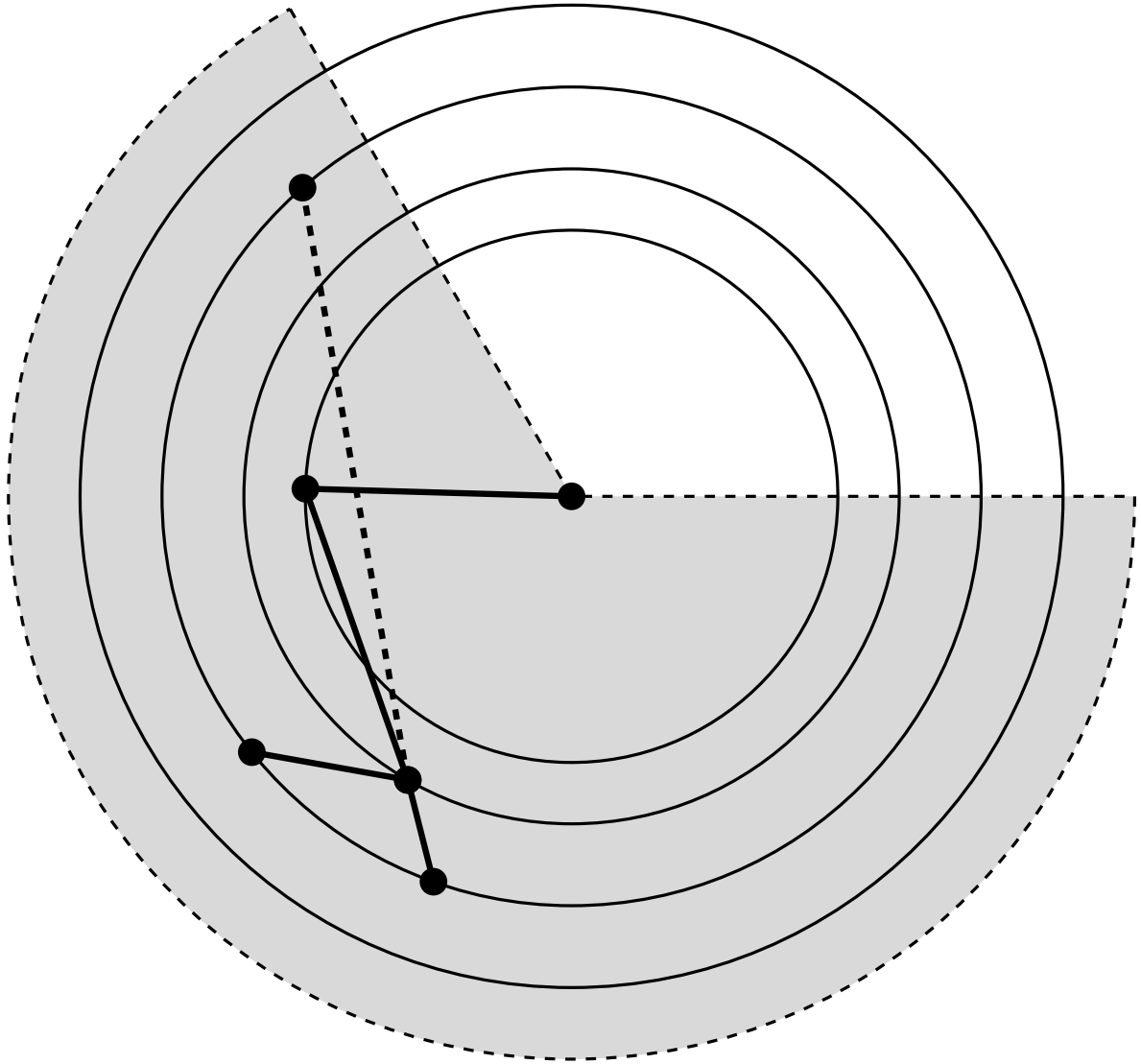


Beispiel Radiallayout



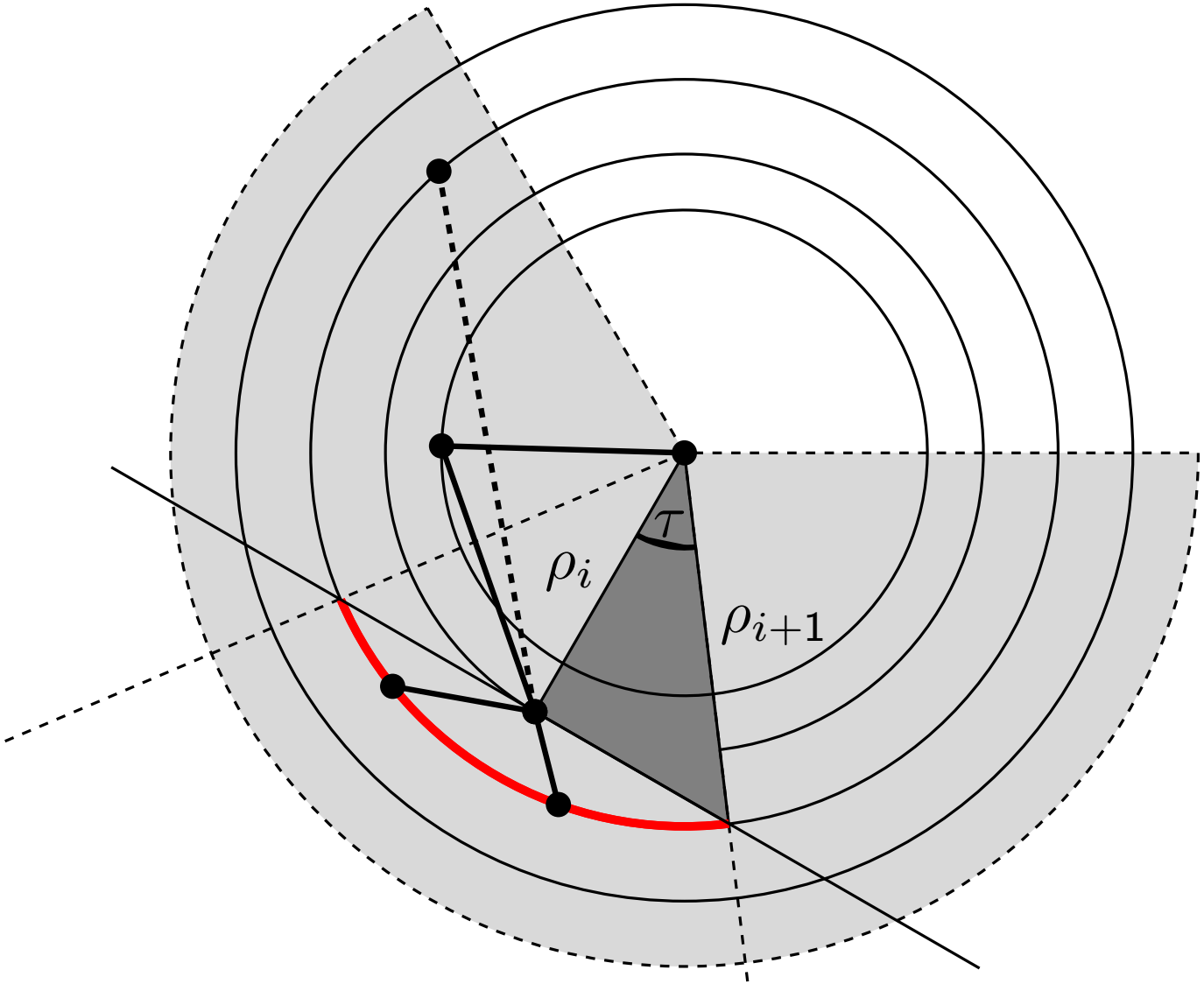


Verlassen des Kreisringsektors



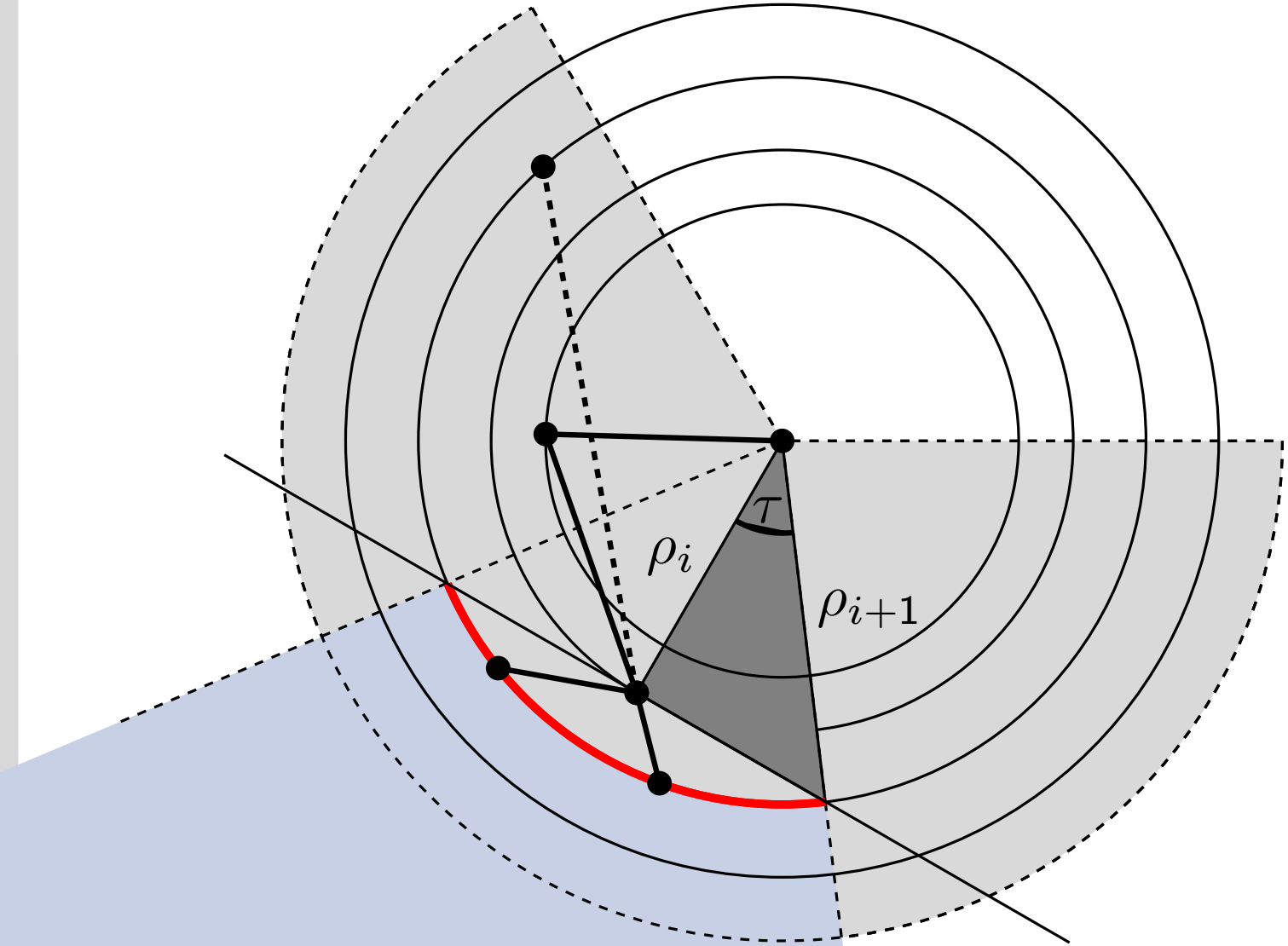
Institute for Theoretical Informatics
Algorithmics Group I

Verlassen des Kreisringsektors



$$\cos\tau = \frac{\rho_i}{\rho_{i+1}}$$

Verlassen des Kreisringsektors



$$\cos\tau = \frac{\rho_i}{\rho_{i+1}}$$

Serien-parallele Graphen

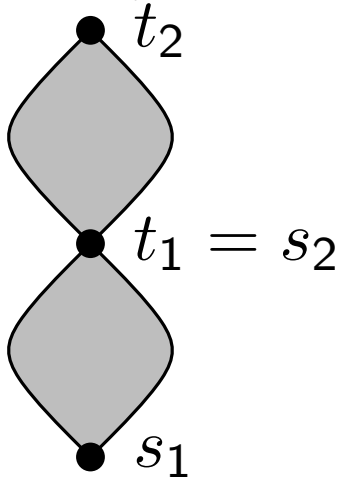
Institute for Theoretical Informatics
Algorithmics Group I

Serien-parallele Graphen

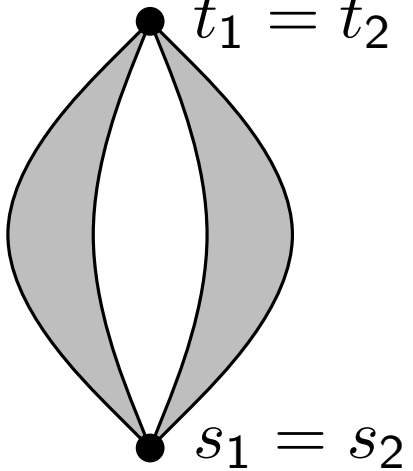
Graph G heißt serien-parallel, wenn er

- aus genau zwei Knoten (Quelle s, Senke t sowie der Kante (s, t) besteht oder
- aus zwei serien-parallelen Graphen G_1, G_2 mit Quellen s_1, s_2 und Senken t_1, t_2 durch eine der folgenden Kombinationen hervorgeht

serielle Komposition:
Identifiziere t_1 und s_2,
s_1 neue Quelle, T_2 neue Senke



parallele Komposition:
Identifiziere s_1, s_2 als neue Quelle
Identifiziere t_1, t_2 als neue Senke



Serien-parallele Graphen II

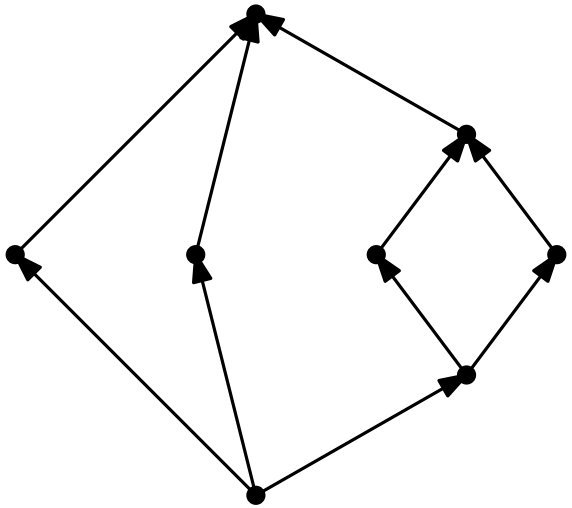
Lemma

Serien-parallele Graphen sind azyklisch und planar.

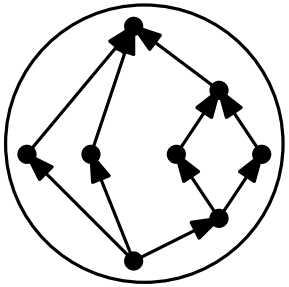
Beschreibung des rekursiven Aufbaus durch binären Baum:

- Blätter sind Kanten (Q-Knoten)
- Innere Knoten sind S- oder P-Knoten

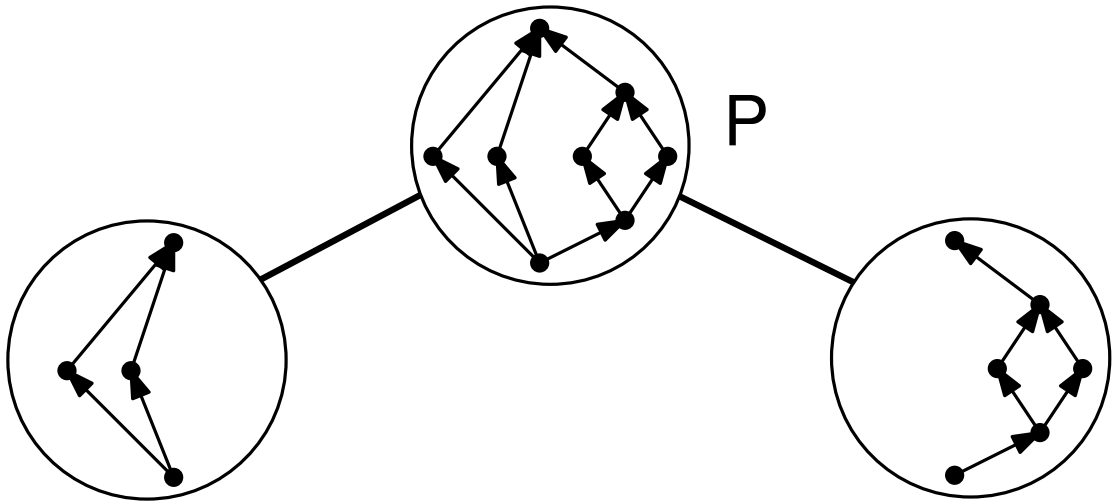
(vgl. SPQR-Baum)



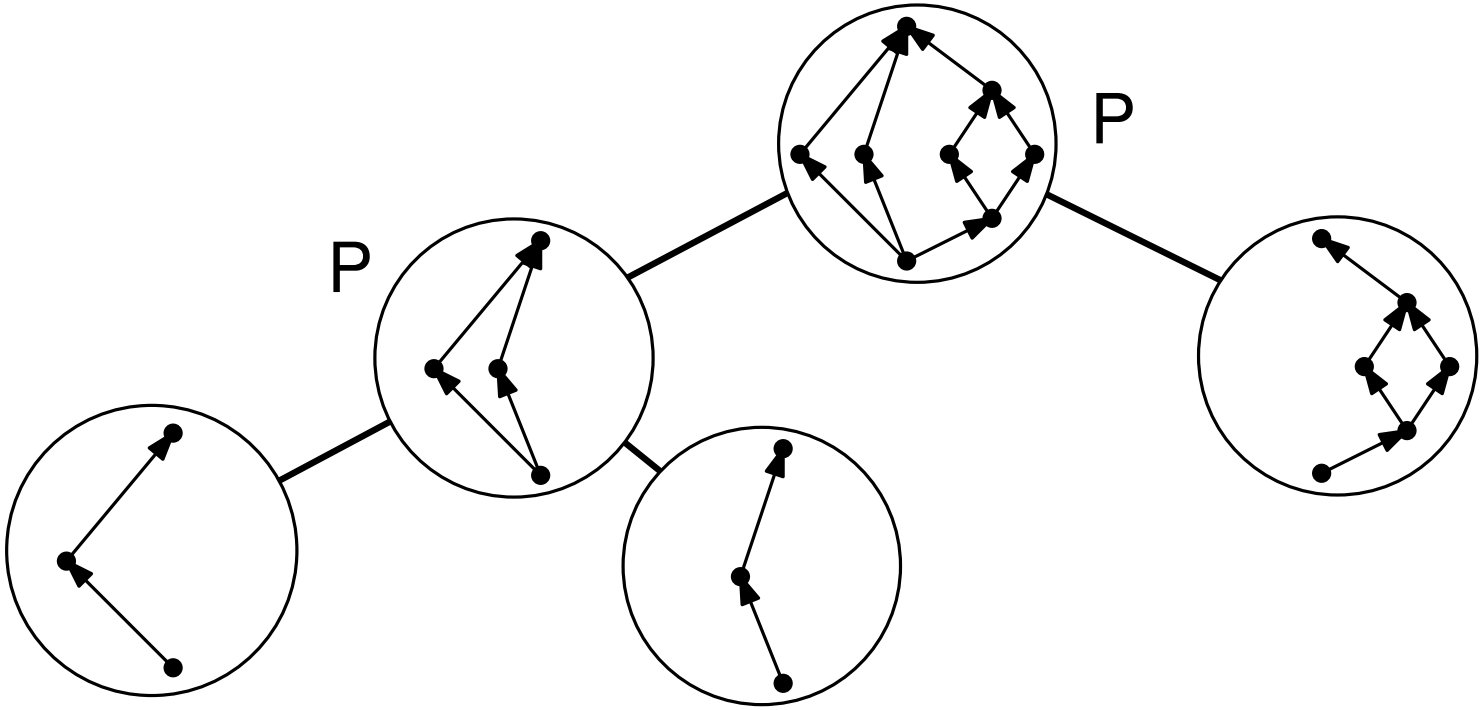
SP-Graphen: Dekompositionsbaum



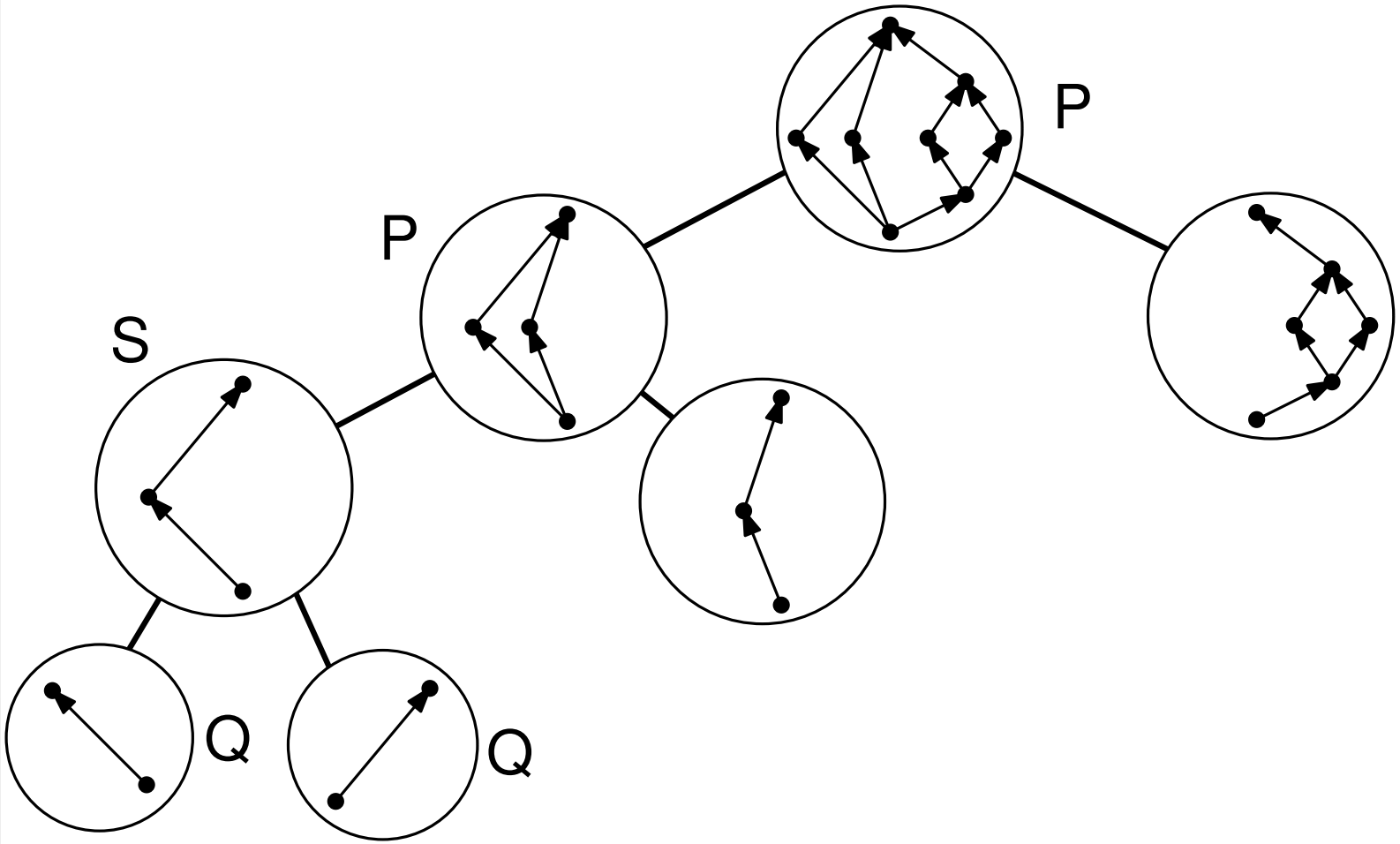
SP-Graphen: Dekompositionsbaum



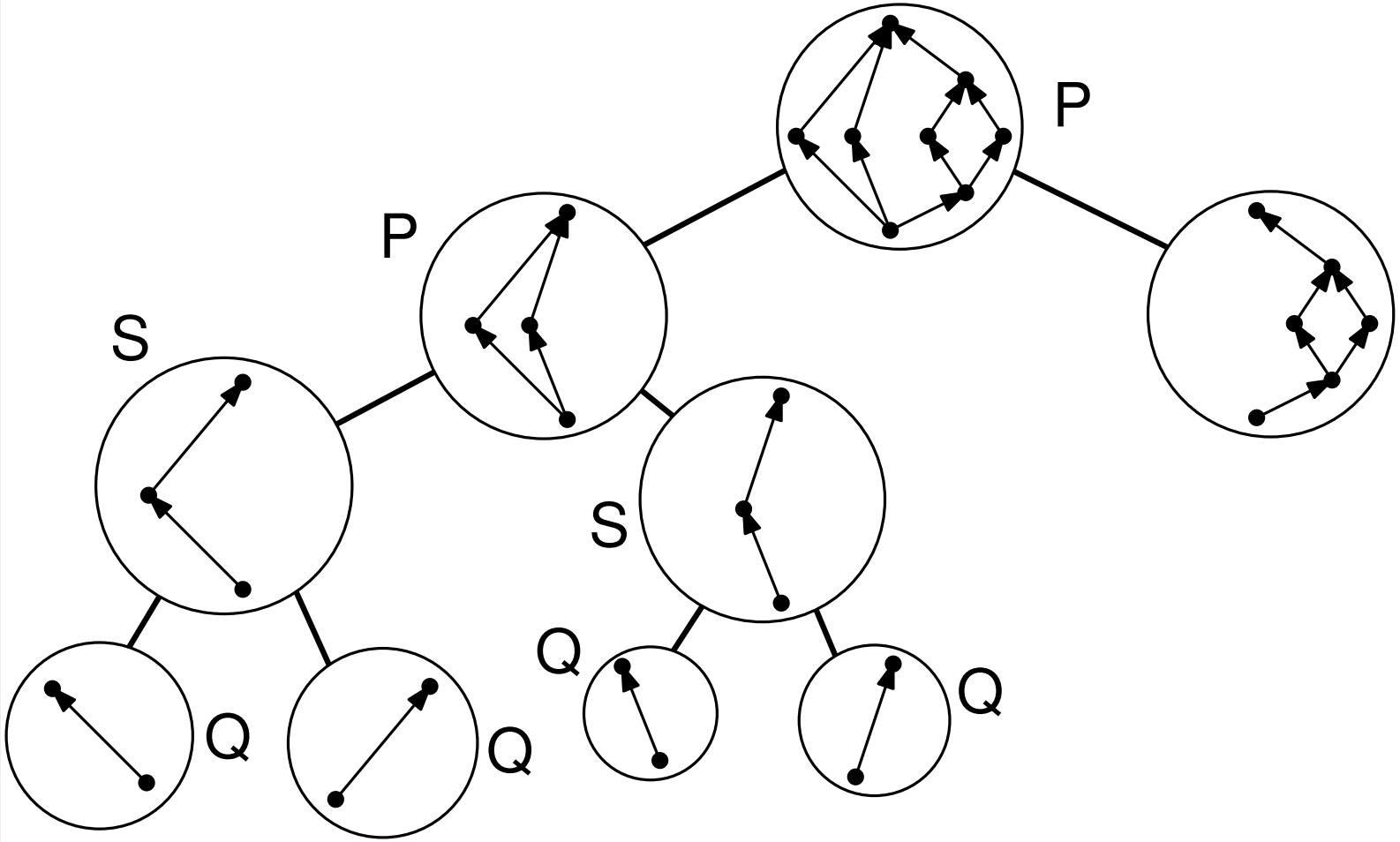
SP-Graphen: Dekompositionsbaum



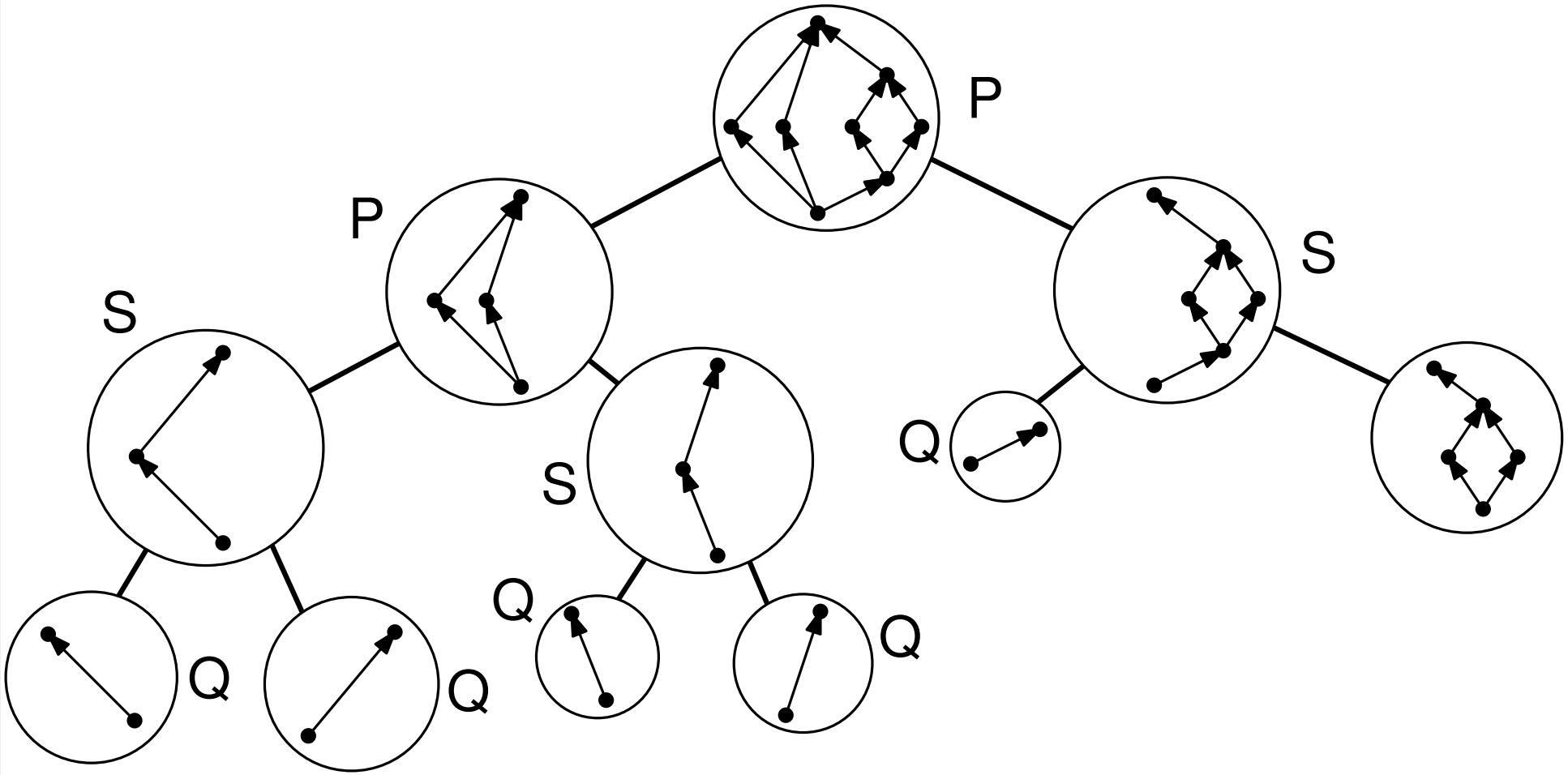
SP-Graphen: Dekompositionsbaum



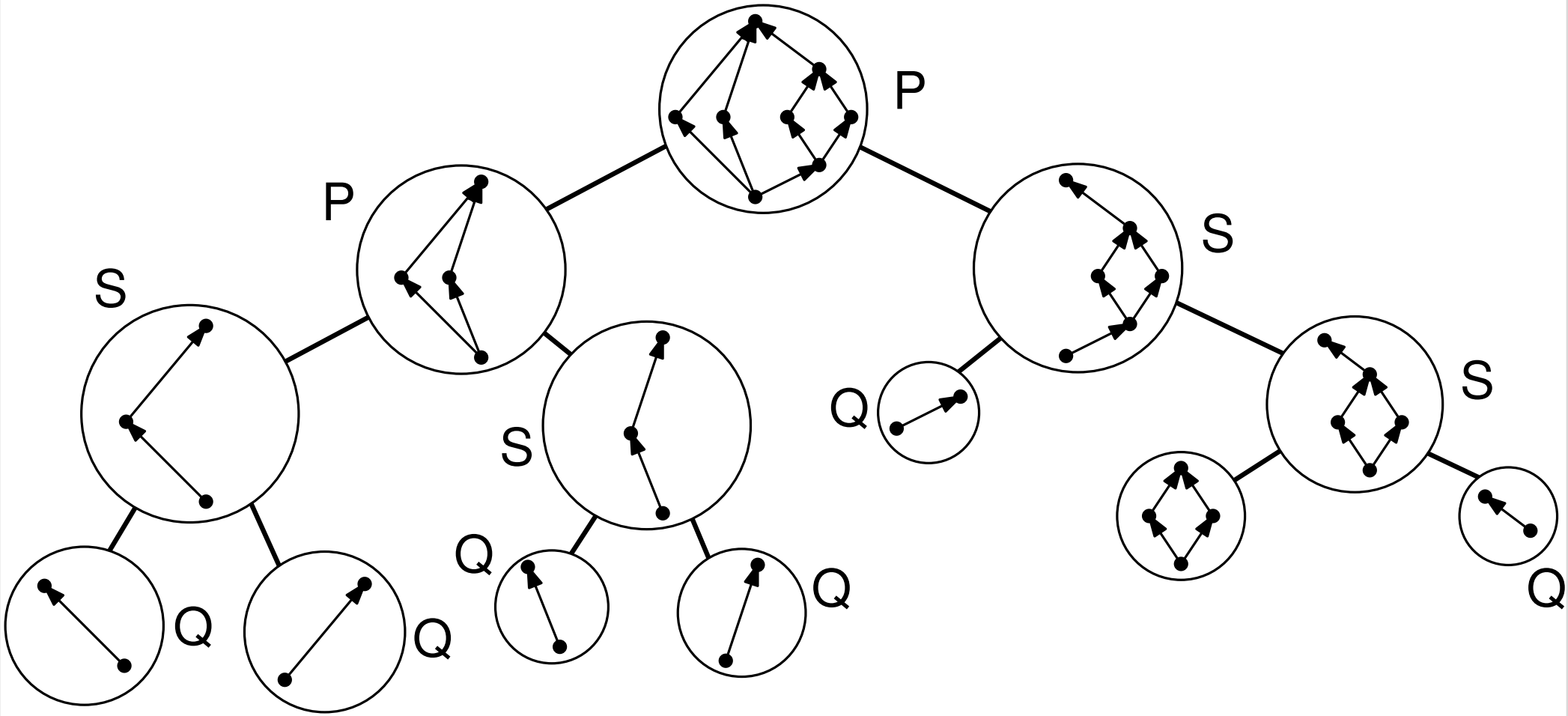
SP-Graphen: Dekompositionsbaum



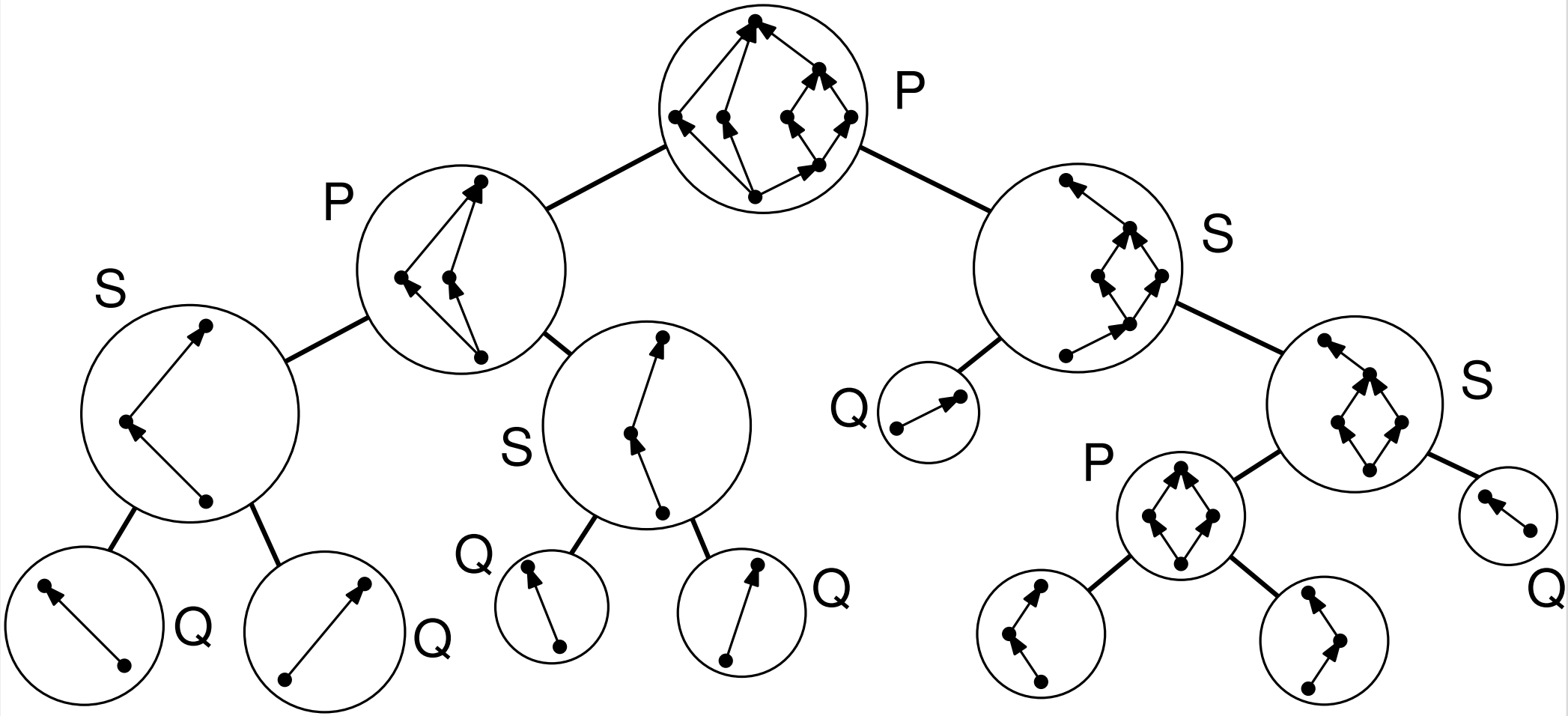
SP-Graphen: Dekompositionsbaum



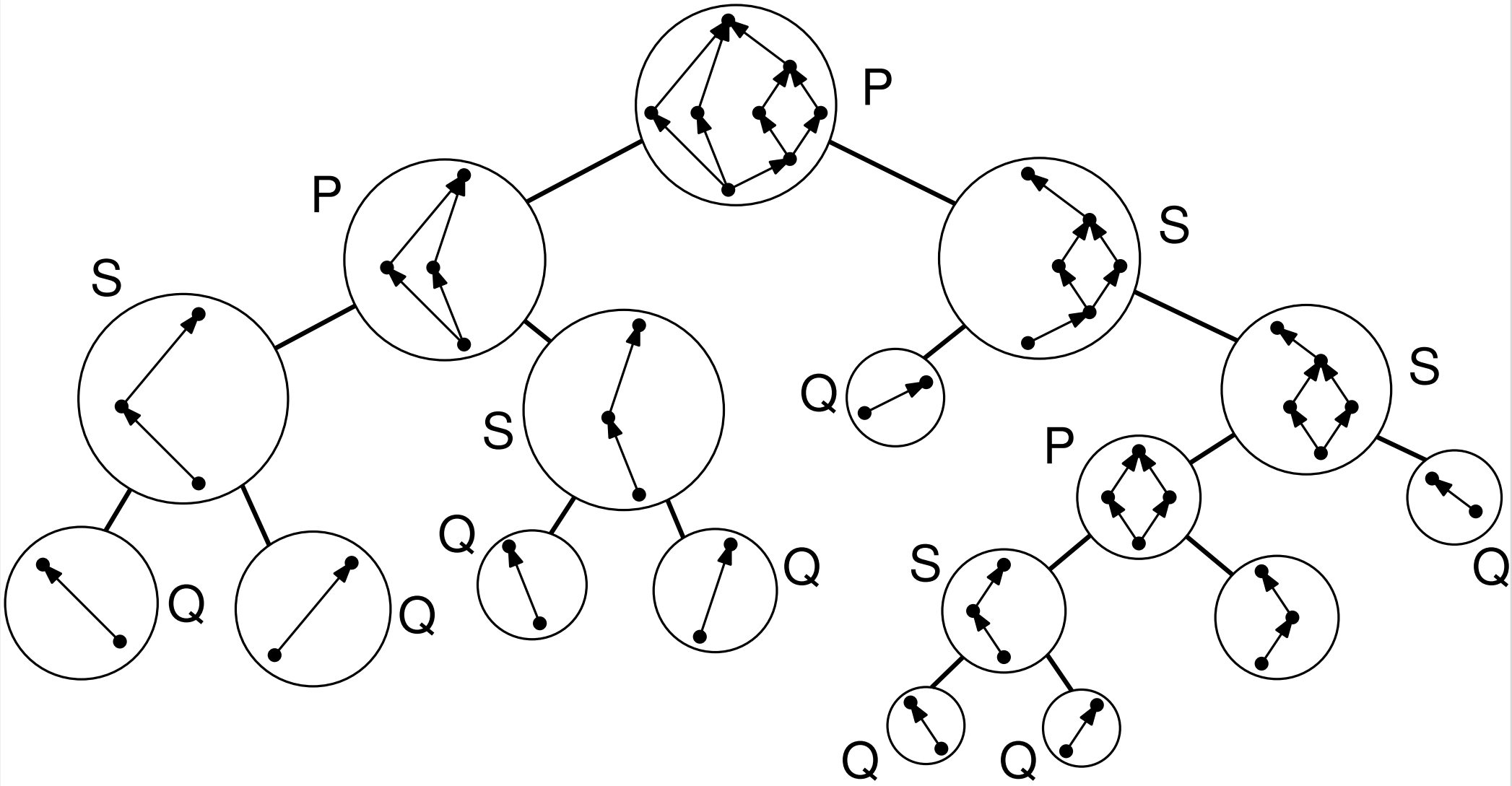
SP-Graphen: Dekompositionsbaum



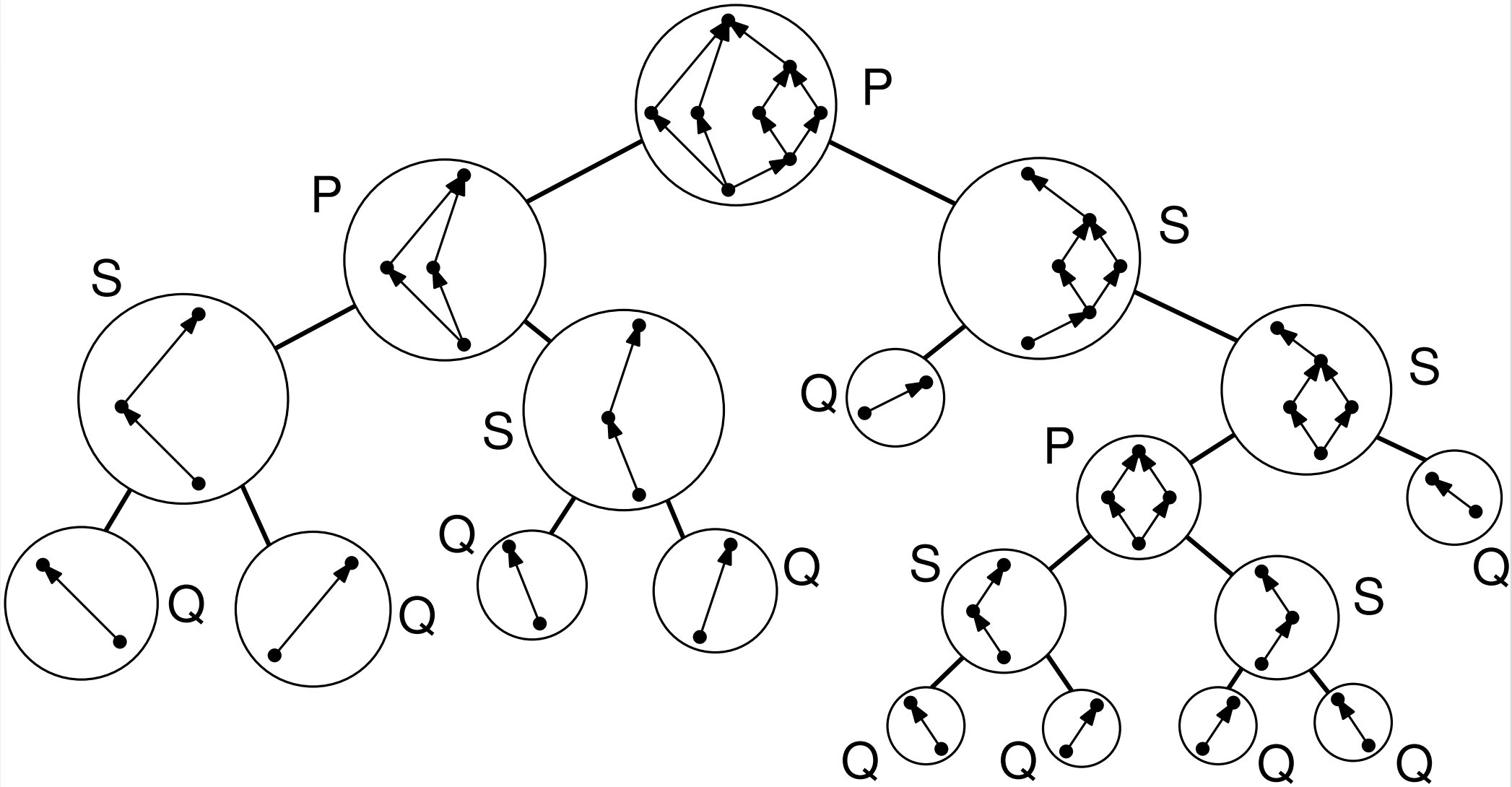
SP-Graphen: Dekompositionsbaum



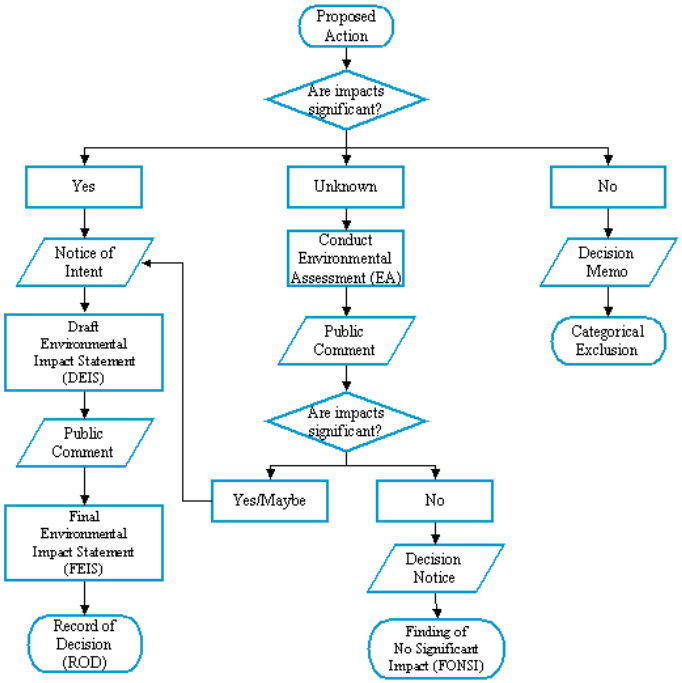
SP-Graphen: Dekompositionsbaum



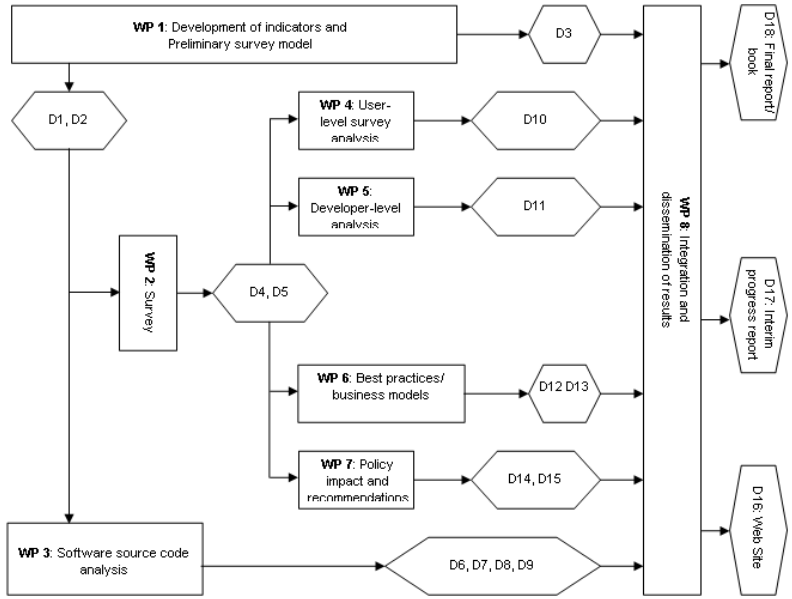
SP-Graphen: Dekompositionsbaum



SP-Graphen in Anwendungen



Ablaufdiagramme



PERT-Diagramme

(Program Evaluation and Review Technique)

Außerdem: Linearzeitalgorithmen für sonst NP-vollständige Probleme (z.B. Maximum Independent Set)

Untere Schranke für die Fläche

Satz

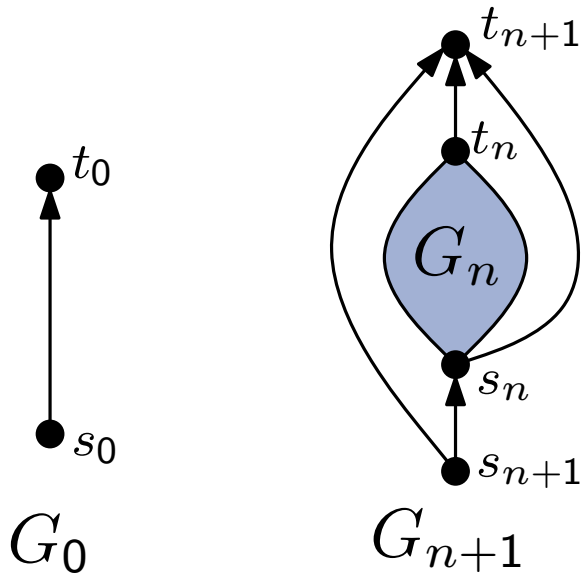
Jedes kreuzungsfreie Aufwärtslayout für geordnete einfache serien-parallele Graphen mit n Knoten benbötigt im worst case ein Gitter der Größe $\Omega(2^n)$.

Untere Schranke für die Fläche

Satz

Jedes kreuzungsfreie Aufwärtslayout für geordnete einfache serien-parallele Graphen mit n Knoten benbötigt im worst case ein Gitter der Größe $\Omega(2^n)$.

Beweis:

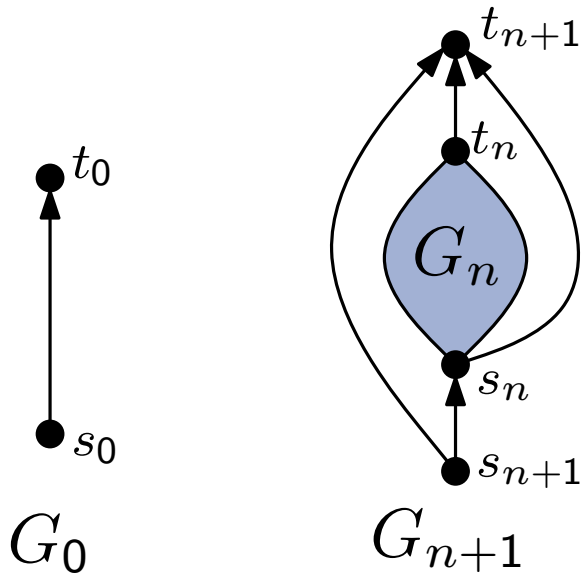


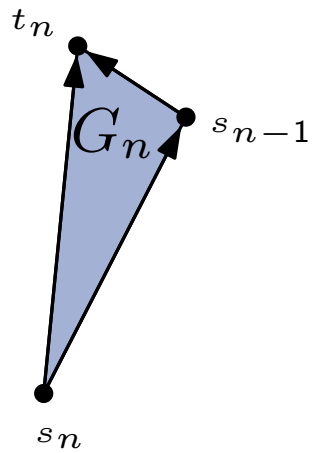
Untere Schranke für die Fläche

Satz

Jedes kreuzungsfreie Aufwärtslayout für geordnete einfache serien-parallele Graphen mit n Knoten benbötigt im worst case ein Gitter der Größe $\Omega(2^n)$.

Beweis:



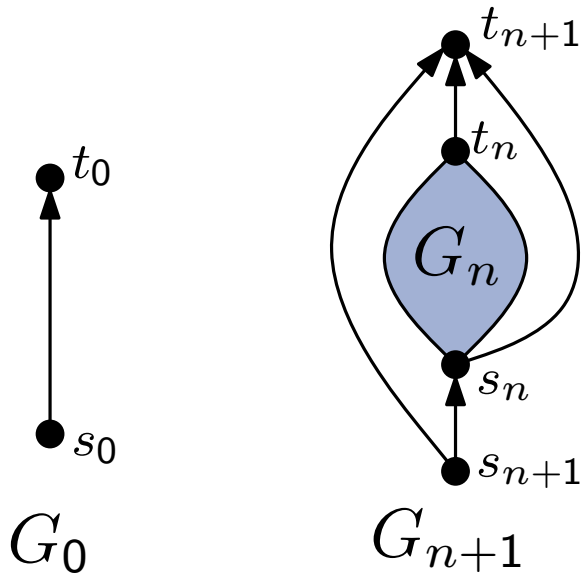


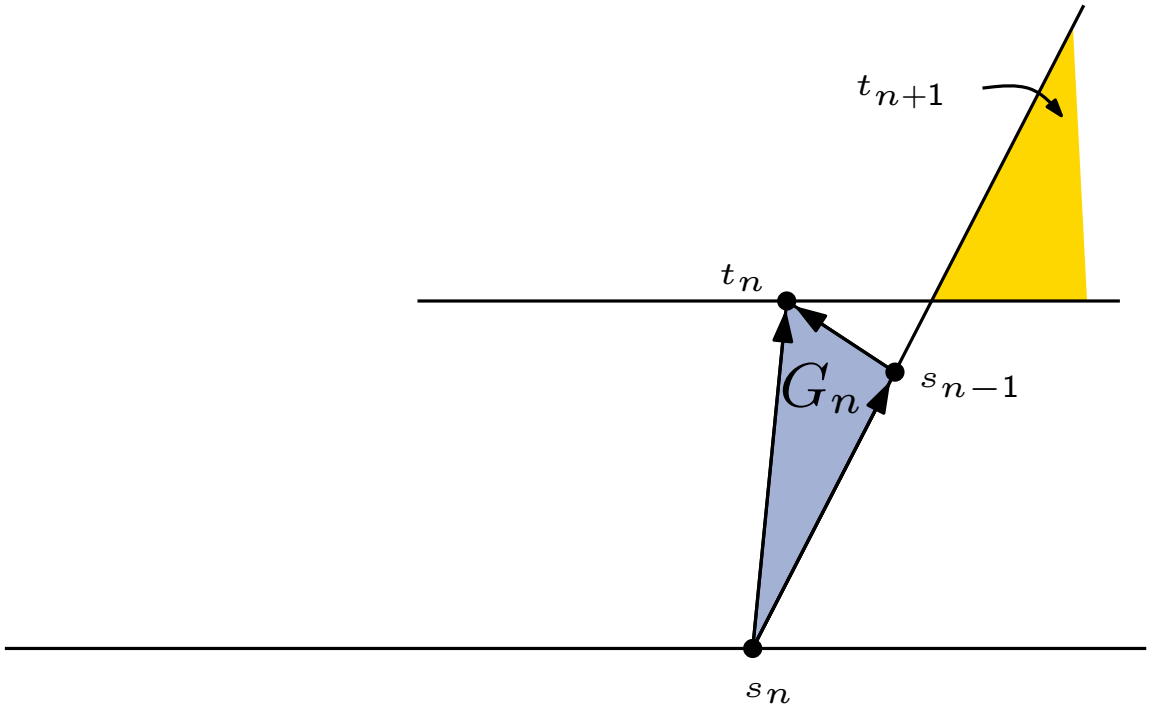
Untere Schranke für die Fläche

Satz

Jedes kreuzungsfreie Aufwärtslayout für geordnete einfache serien-parallele Graphen mit n Knoten benbötigt im worst case ein Gitter der Größe $\Omega(2^n)$.

Beweis:



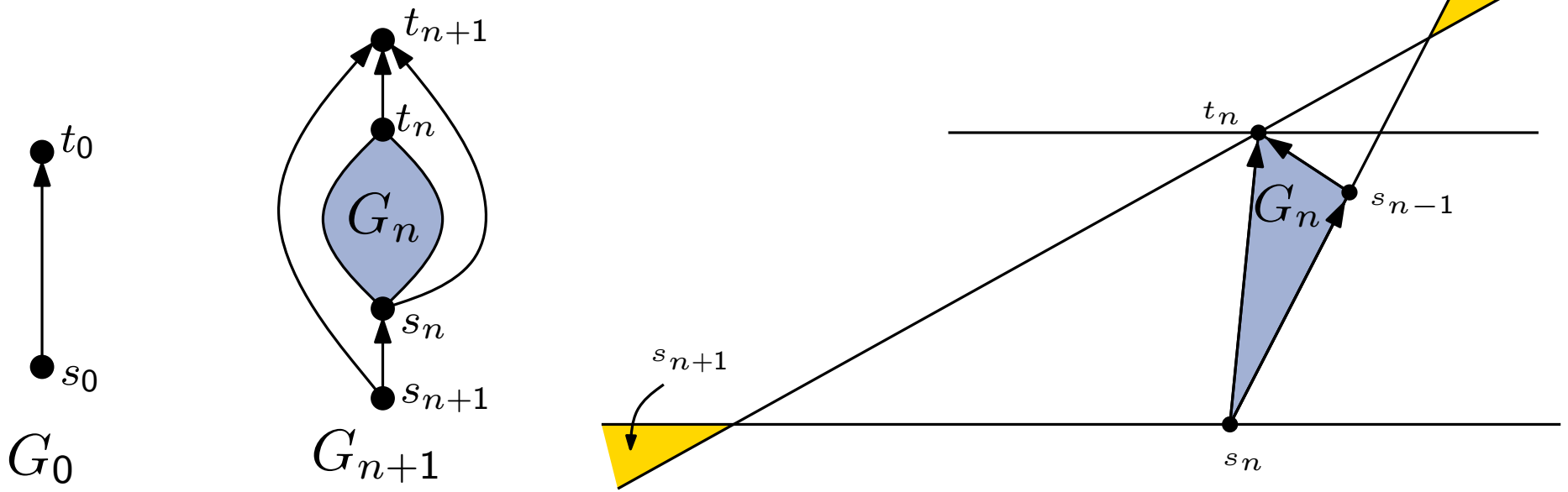


Untere Schranke für die Fläche

Satz

Jedes kreuzungsfreie Aufwärtslayout für geordnete einfache serien-parallele Graphen mit n Knoten benbötigt im worst case ein Gitter der Größe $\Omega(2^n)$.

Beweis:

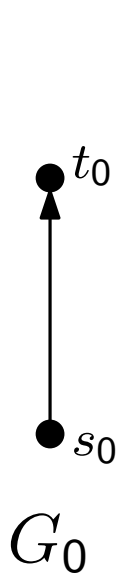


Untere Schranke für die Fläche

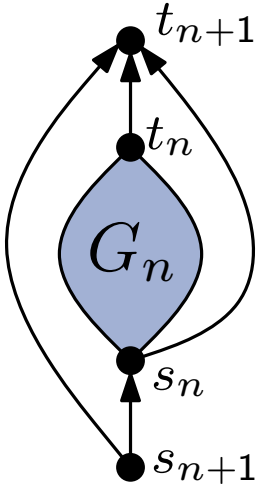
Satz

Jedes kreuzungsfreie Aufwärtslayout für geordnete einfache serien-parallele Graphen mit n Knoten benbötigt im worst case ein Gitter der Größe $\Omega(2^n)$.

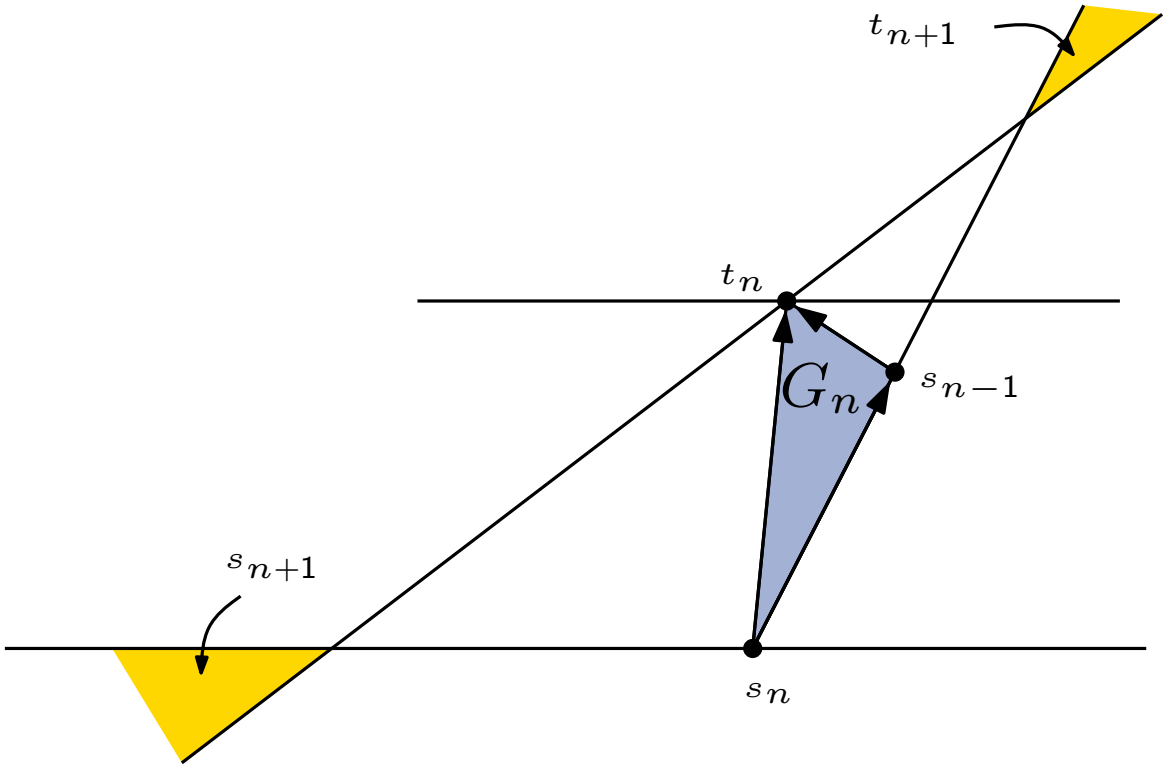
Beweis:



G_0



G_{n+1}

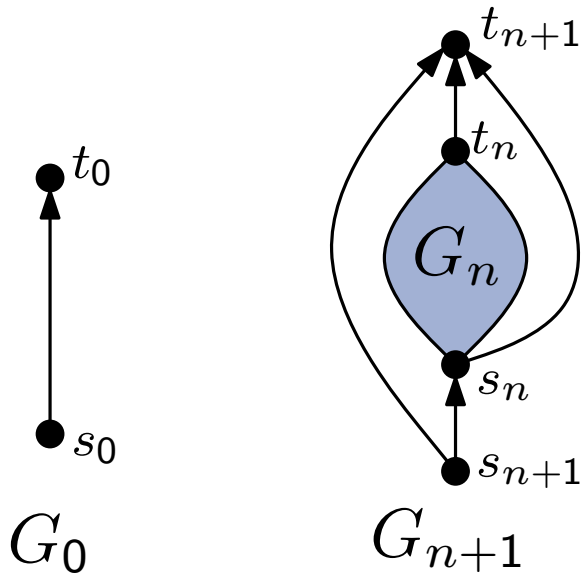


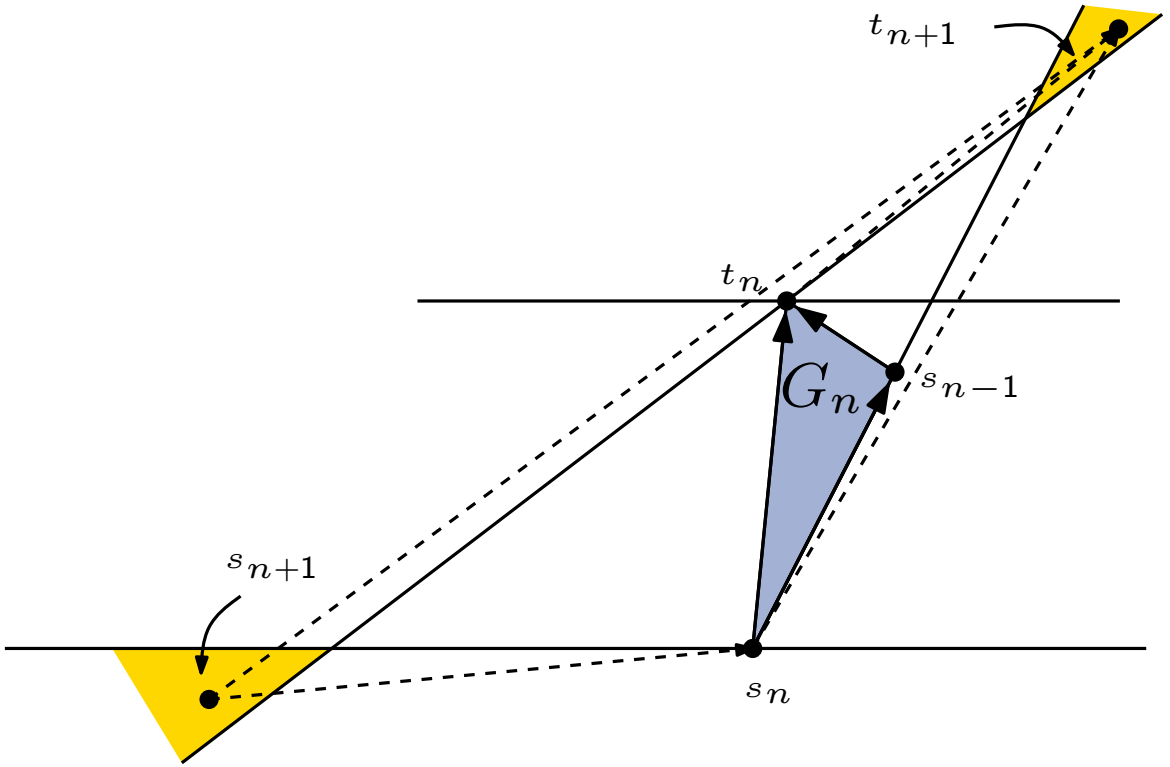
Untere Schranke für die Fläche

Satz

Jedes kreuzungsfreie Aufwärtslayout für geordnete einfache serien-parallele Graphen mit n Knoten benbötigt im worst case ein Gitter der Größe $\Omega(2^n)$.

Beweis:



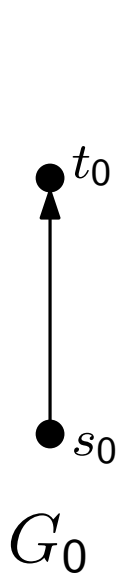


Untere Schranke für die Fläche

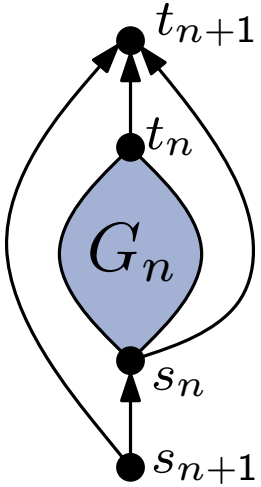
Satz

Jedes kreuzungsfreie Aufwärtslayout für geordnete einfache serien-parallele Graphen mit n Knoten benbötigt im worst case ein Gitter der Größe $\Omega(2^n)$.

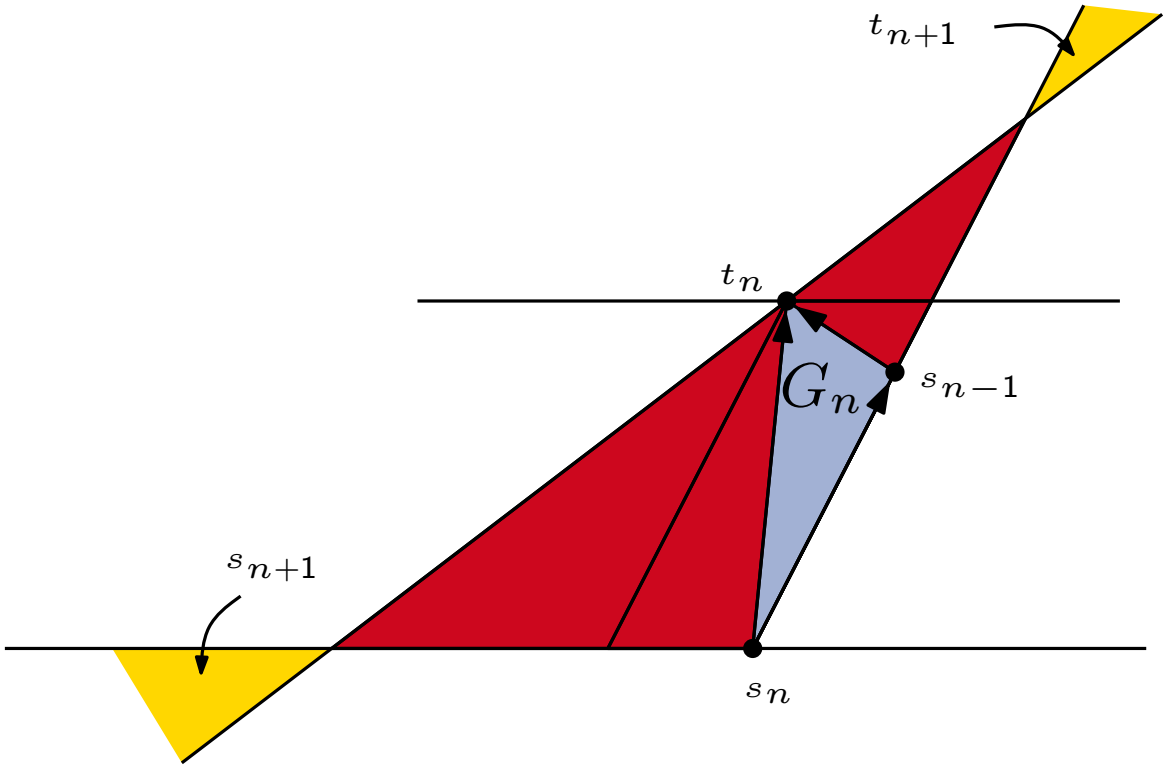
Beweis:



G_0



G_{n+1}

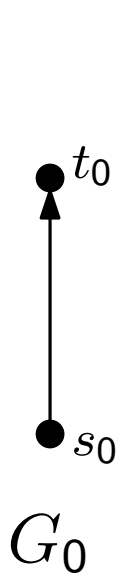


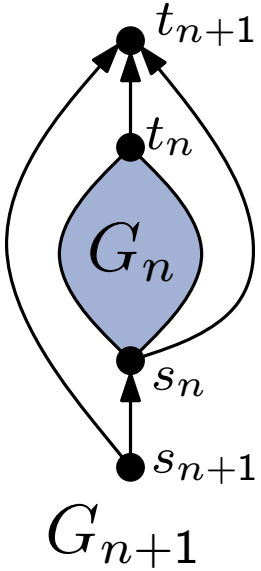
Untere Schranke für die Fläche

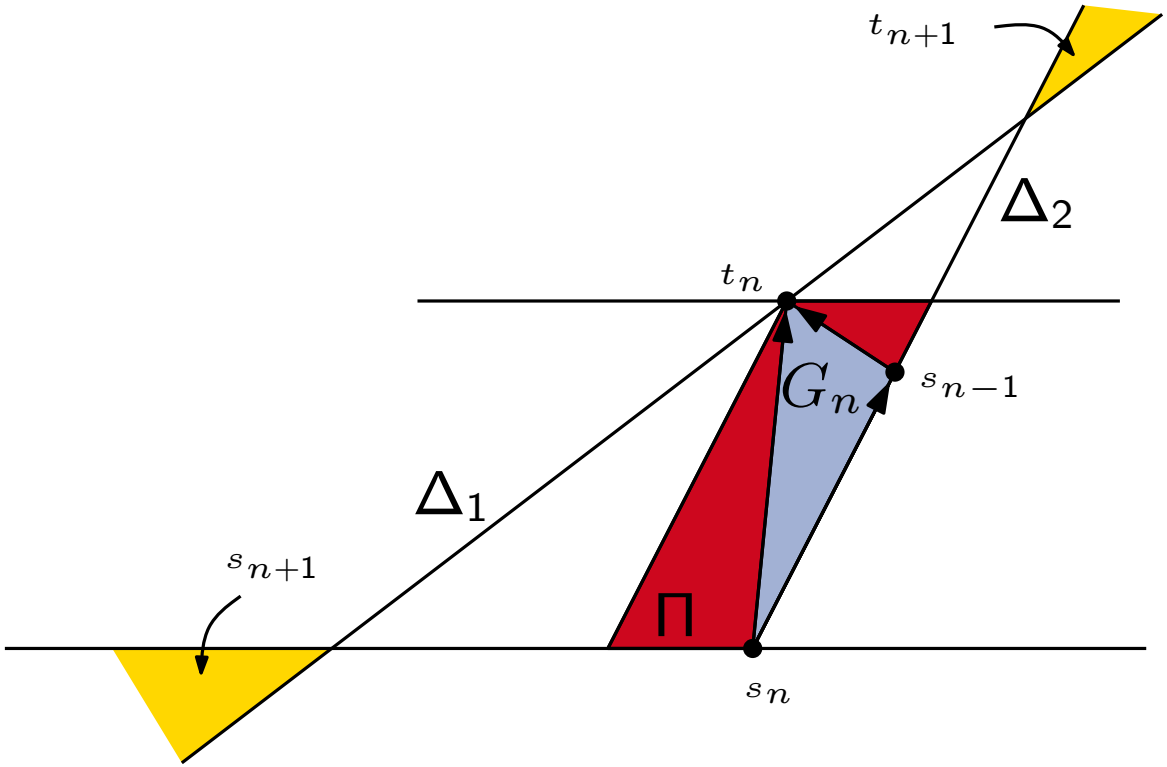
Satz

Jedes kreuzungsfreie Aufwärtslayout für geordnete einfache serien-parallele Graphen mit n Knoten benbötigt im worst case ein Gitter der Größe $\Omega(2^n)$.

Beweis:





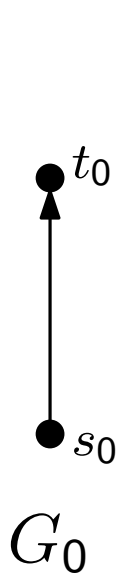


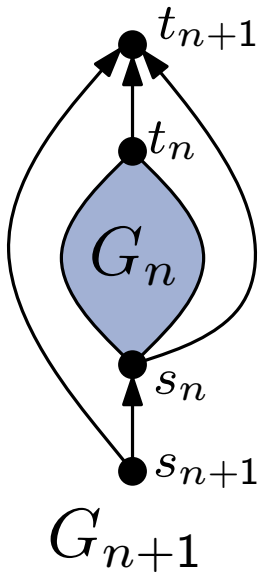
Untere Schranke für die Fläche

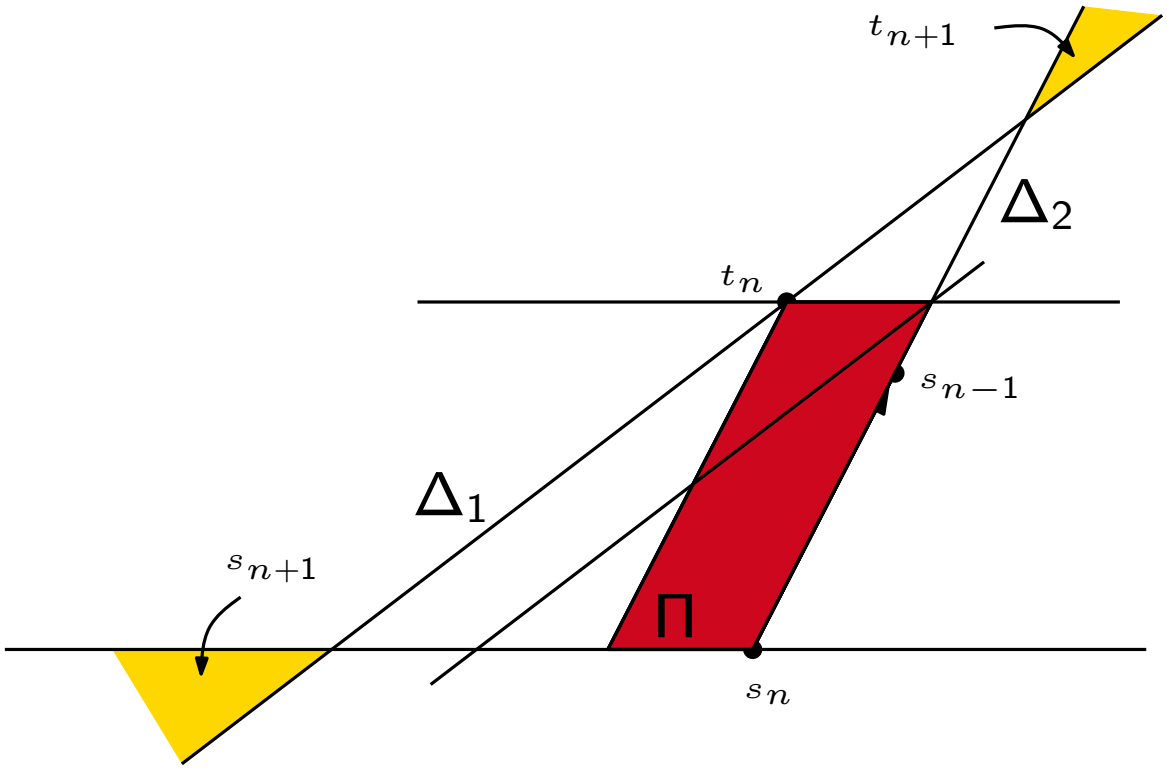
Satz

Jedes kreuzungsfreie Aufwärtslayout für geordnete einfache serien-parallele Graphen mit n Knoten benbötigt im worst case ein Gitter der Größe $\Omega(2^n)$.

Beweis:





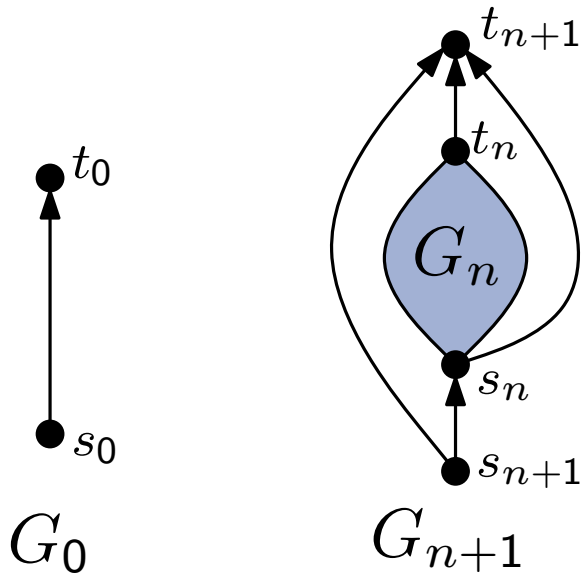


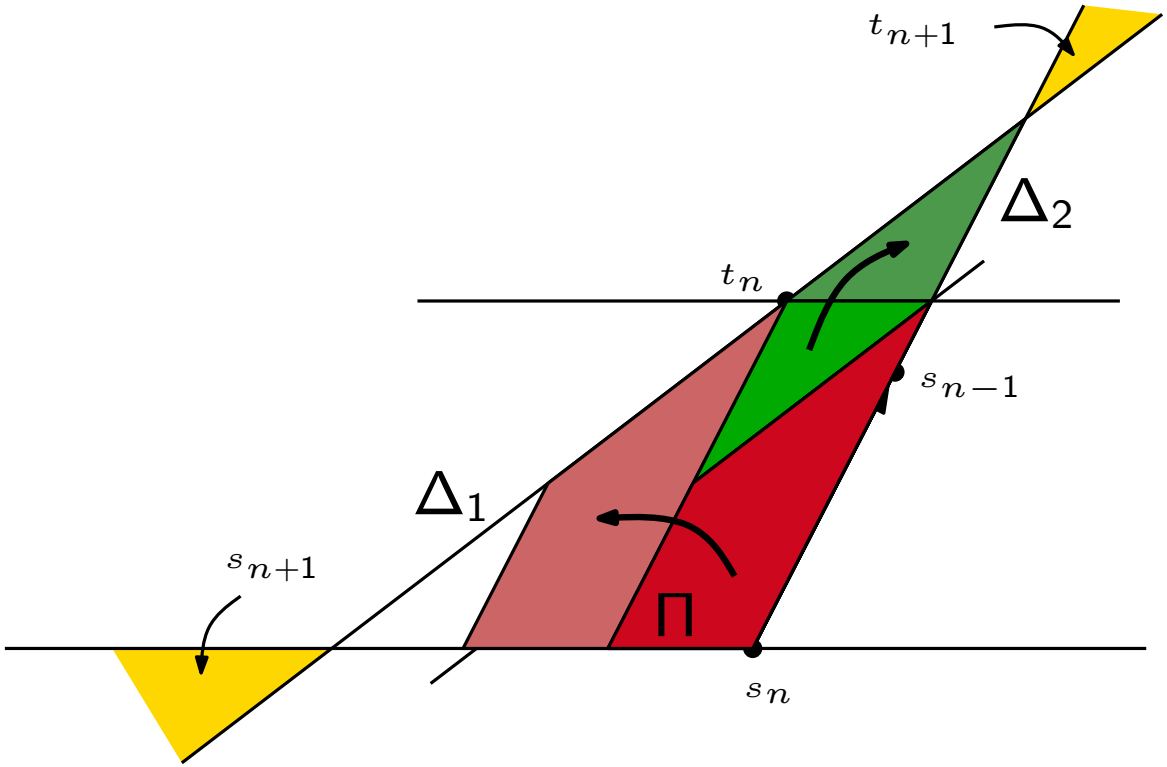
Untere Schranke für die Fläche

Satz

Jedes kreuzungsfreie Aufwärtslayout für geordnete einfache serien-parallele Graphen mit n Knoten benbötigt im worst case ein Gitter der Größe $\Omega(2^n)$.

Beweis:



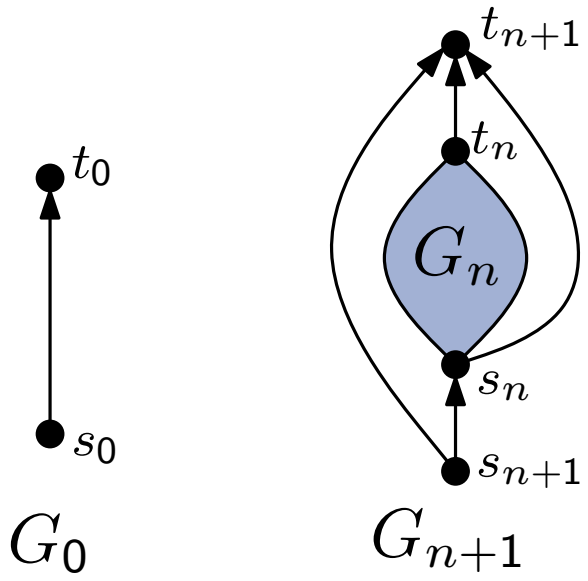


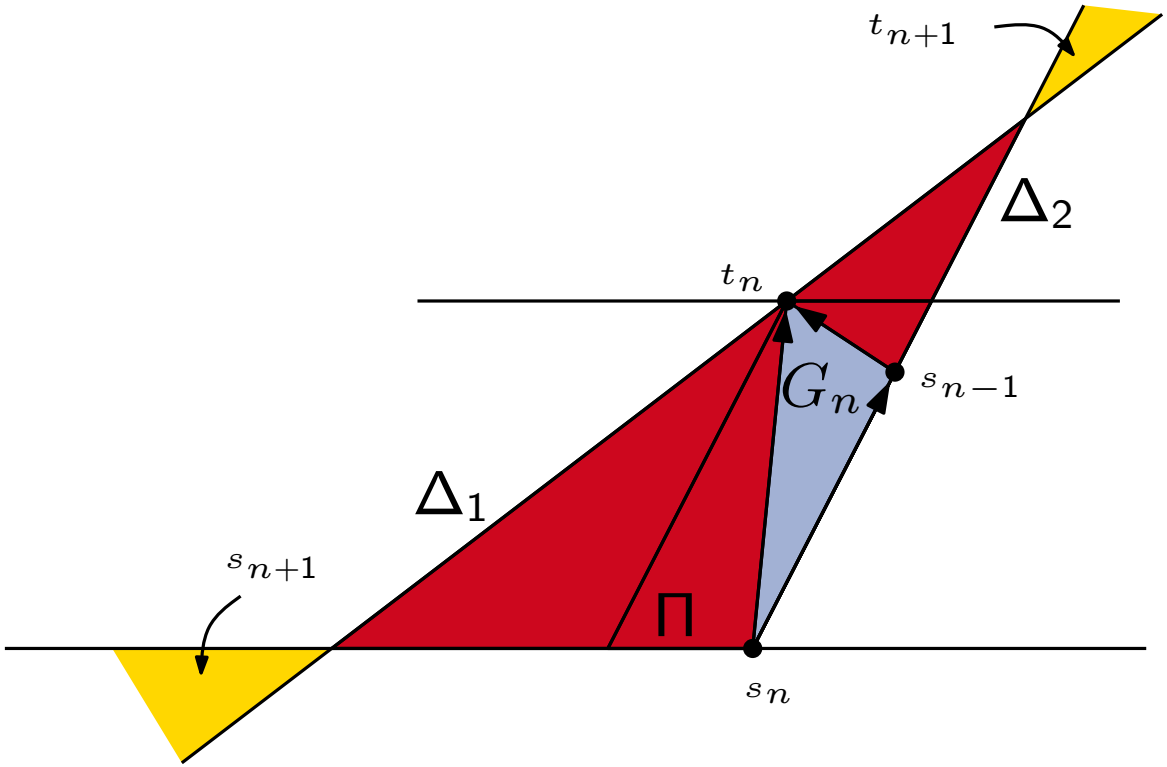
Untere Schranke für die Fläche

Satz

Jedes kreuzungsfreie Aufwärtslayout für geordnete einfache serien-parallele Graphen mit n Knoten benbötigt im worst case ein Gitter der Größe $\Omega(2^n)$.

Beweis:





Linkslastige Ordnungen

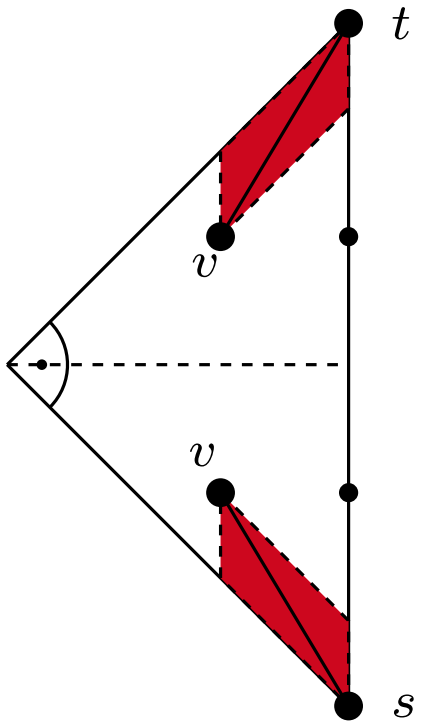
Ordnung heißt linkslastig, wenn Q-Knoten nur als rechte Nachfolger von P-Knoten vorkommen.

Satz

Wenn G serien-parallel, einfach und linkslastig geordnet, so besitzt G Zeichnung der Größe $O(n^2)$.

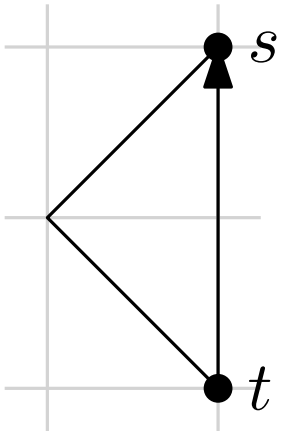
Komponenten des Dekompositionsbaums:

- Layout von G passt in rechtwinkliges, gleichschenkliges Dreieck mit vertikaler Basis, Schenkel nach links.
- Quelle in unterer Ecke, Senke in oberer Ecke, linke Ecke frei
- rechtester Nachbar von $s \neq t$ ($t \neq s$) liegt unterhalb (oberhalb) der Mitte
- v Nachbar der Quelle (Senke) $\Rightarrow$ kein Knoten liegt im Parallelogramm von v und s (t).



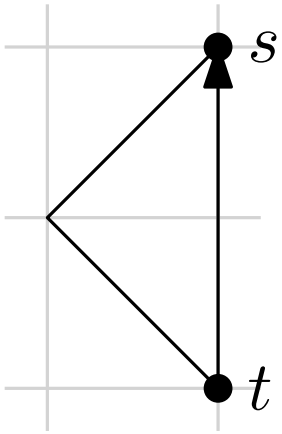
Konstruktion

Q-Knoten (Induktionsanfang):

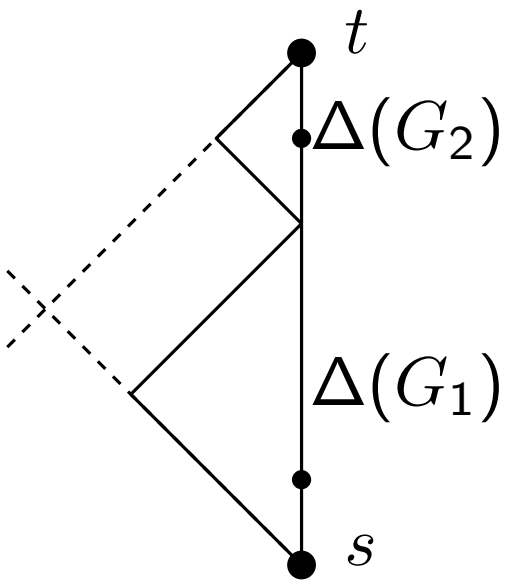


Konstruktion

Q-Knoten (Induktionsanfang):

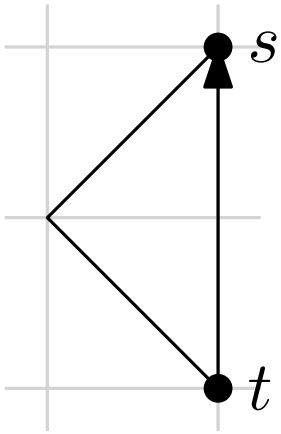


S-Knoten (serielle Komposition):

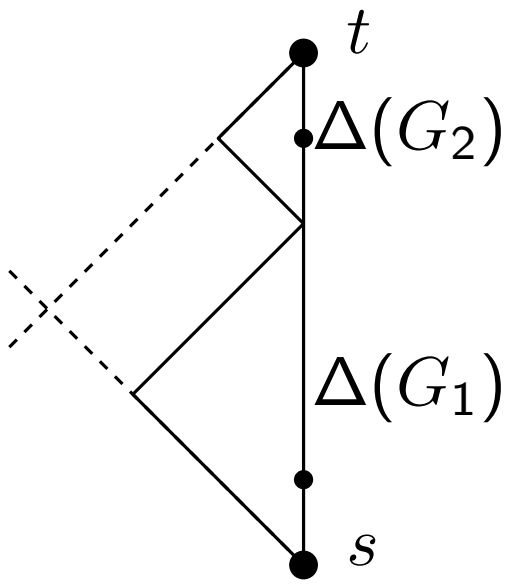


Konstruktion

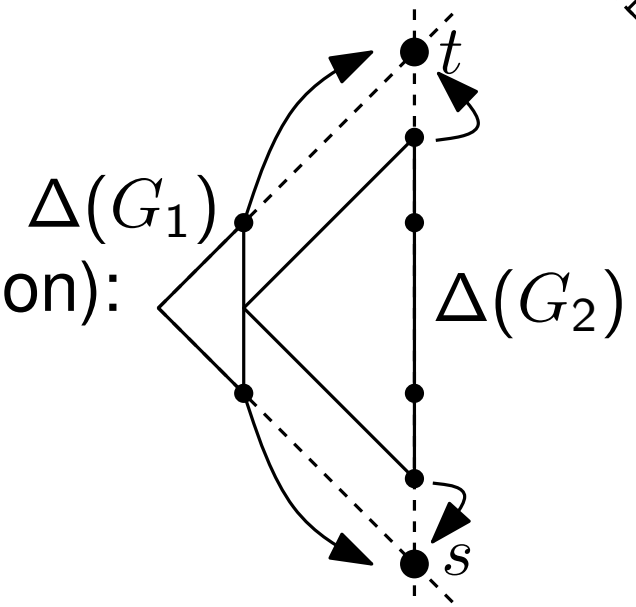
Q-Knoten (Induktionsanfang):



S-Knoten (serielle Komposition):

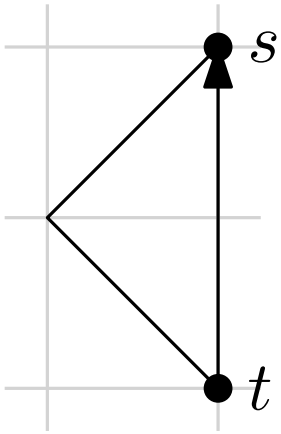


P-Knoten (parallele Komposition):

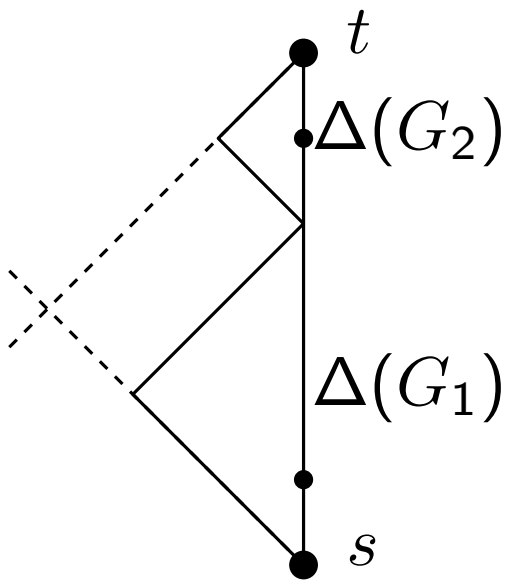


Konstruktion

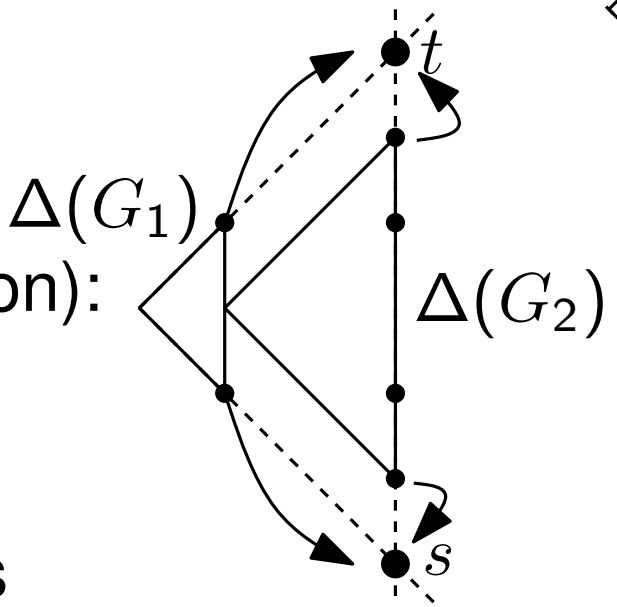
Q-Knoten (Induktionsanfang):



S-Knoten (serielle Komposition):



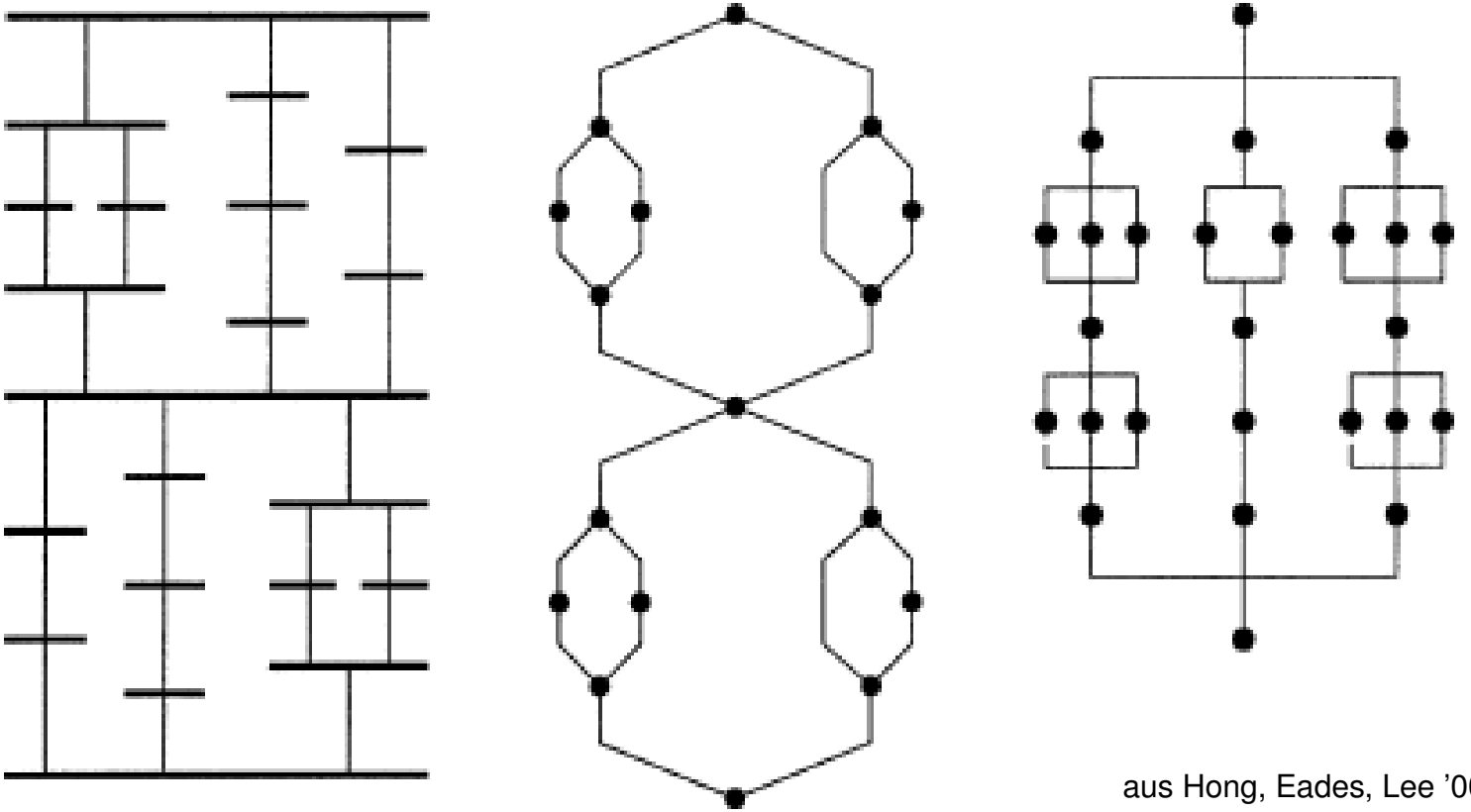
P-Knoten (parallele Komposition):



Eigenschaften:

- kreuzungsfrei und aufwärts
- höchstens quadratische Fläche

Darstellung von Symmetrien in Graphen



aus Hong, Eades, Lee ’00

Darstellung von Symmetrien in Graphen

Definition: Automorphismen eines DAG

Ein Automorphismus eines DAG $G = (V, E)$ ist eine Knotenpermutation $\pi : V \to V$, die Adjazenzen respektiert und alle Kantenrichtungen erhält oder alle Kantenrichtungen umdreht:

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

oder

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

Die Automorphismen von G bilden mit der Hintereinanderausführung eine Gruppe

Darstellung von Symmetrien in Graphen

Definition: Automorphismen eines DAG

Ein Automorphismus eines DAG $G = (V, E)$ ist eine Knotenpermutation $\pi : V \to V$, die Adjazenzen respektiert und alle Kantenrichtungen erhält oder alle Kantenrichtungen umdreht:

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

oder

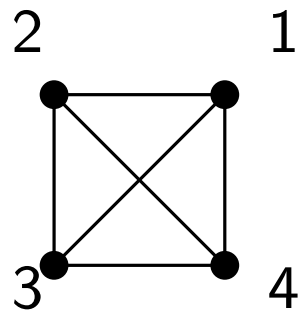
$$(u, v) \in E \Leftrightarrow (\pi(v), \pi(u)) \in E.$$

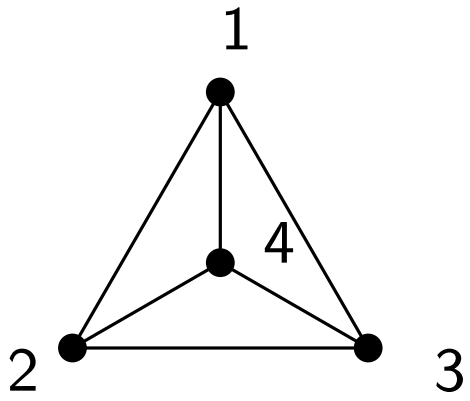
Die Automorphismen von G bilden mit der Hintereinanderausführung eine Gruppe.

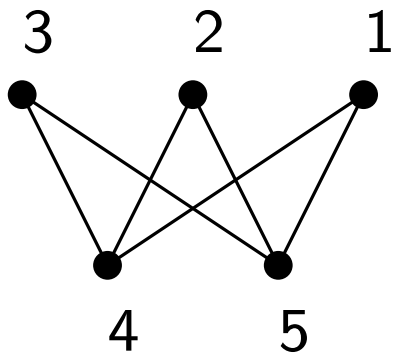
- die Automorphismengruppe eines Graphen zu bestimmen ist GI-vollständig
- für Graphen mit beschränktem Maximalgrad und planare Graphen geht das in Polynomialzeit

Darstellbare Symmetrien

- ein Automorphismus π eines Graphen ist darstellbar, wenn es eine Zeichnung gibt, die eine Symmetrie enthält, welche π induziert
- Lin charakterisiert darstellbare Automorphismen [Lin '92]
- Darstellbare Automorphismen eines Graphen zu finden ist NP-schwer
- für planare Graphen ist das wieder in Polynomialzeit möglich

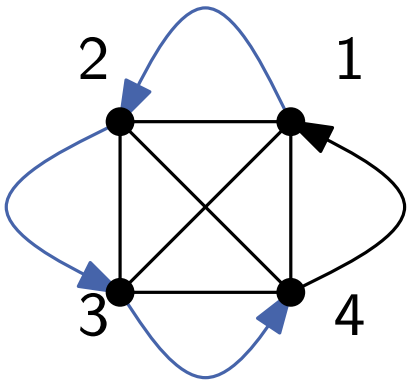




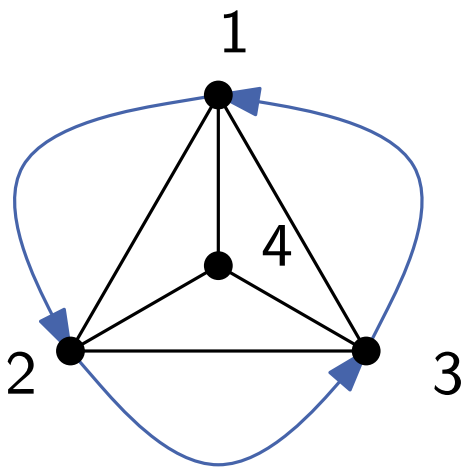


Darstellbare Symmetrien

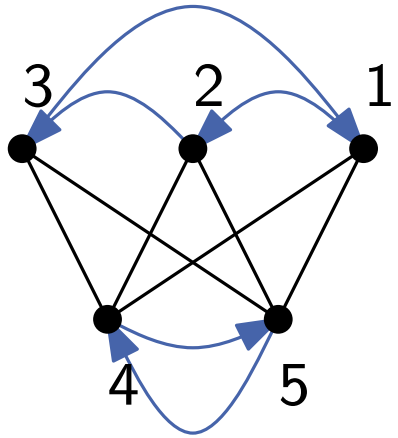
- ein Automorphismus π eines Graphen ist darstellbar, wenn es eine Zeichnung gibt, die eine Symmetrie enthält, welche π induziert
- Lin charakterisiert darstellbare Automorphismen [Lin '92]
- Darstellbare Automorphismen eines Graphen zu finden ist NP-schwer
- für planare Graphen ist das wieder in Polynomialzeit möglich



stellt $1 \to 2 \to 3 \to 4 \to 1$ dar,
aber nicht $1 \to 2 \to 3 \to 1$

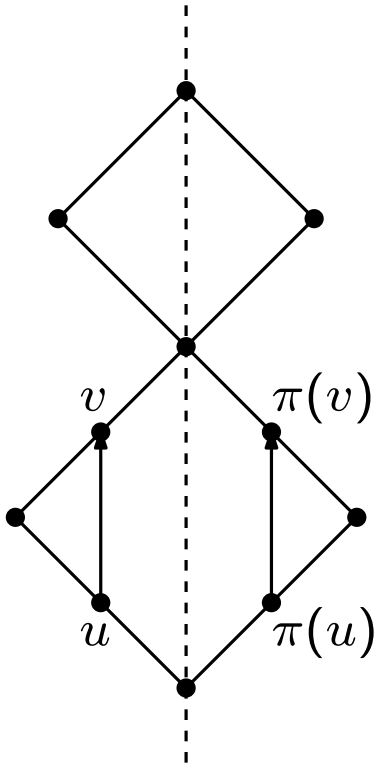


stellt $1 \to 2 \to 3 \to 1$ dar, aber
nicht $1 \to 2 \to 3 \to 4 \to 1$



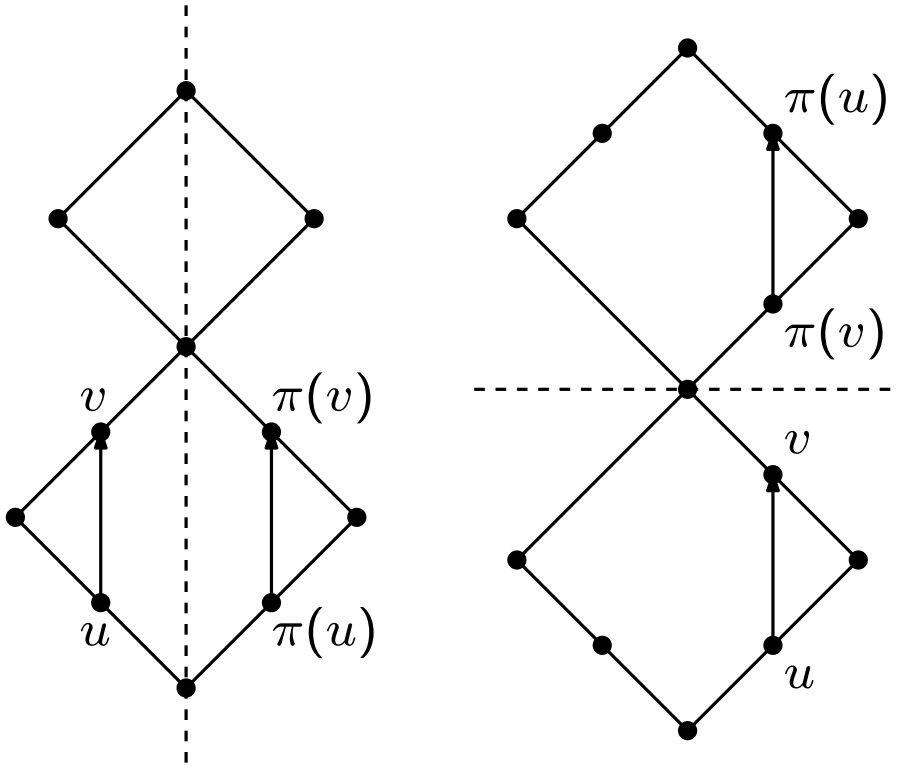
$1 \to 2 \to 3 \to 1, 4 \to 5 \to 4$
nicht darstellbar

Symmetrien in SP-Graphen



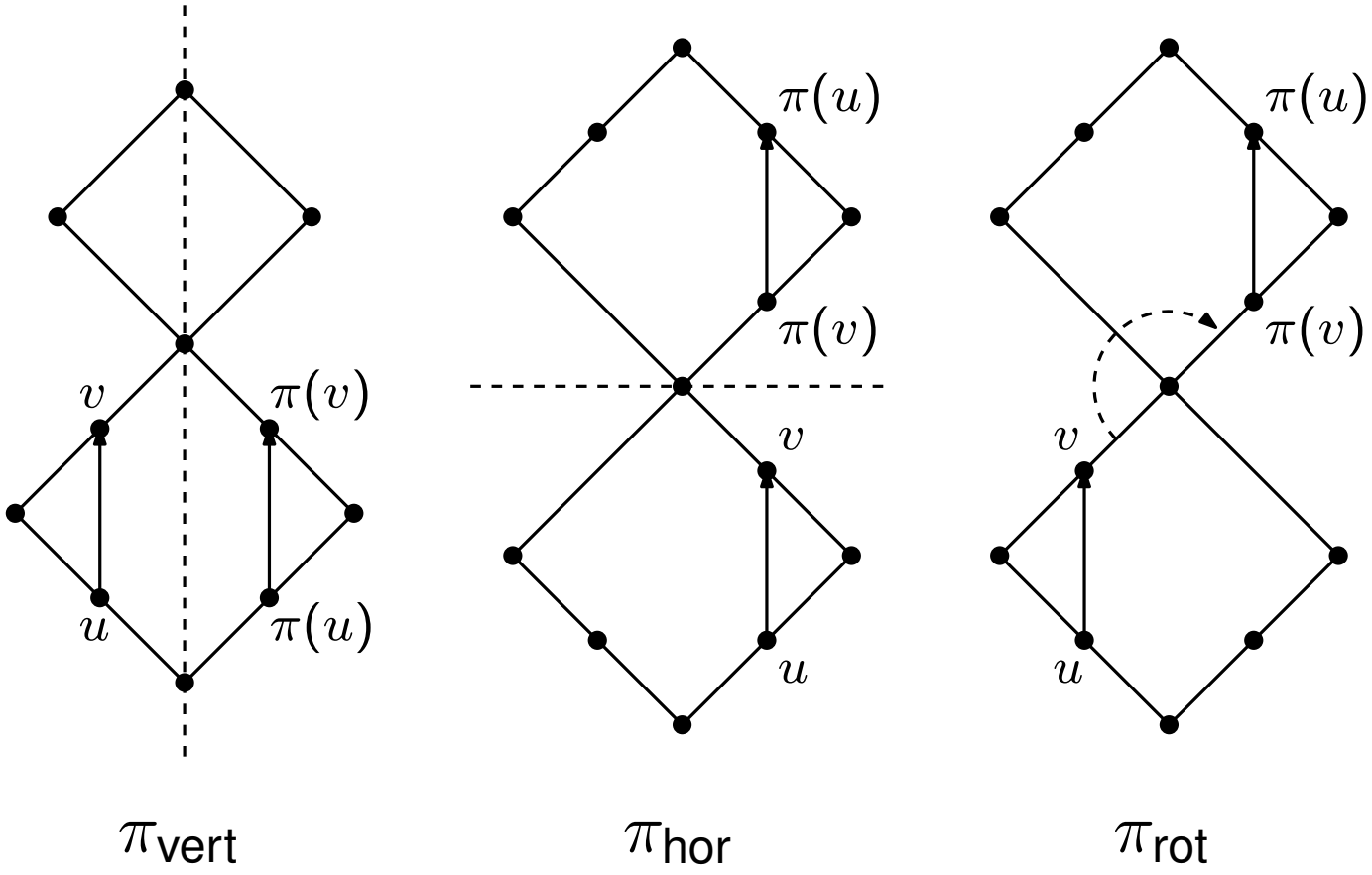
π_{vert}

Symmetrien in SP-Graphen



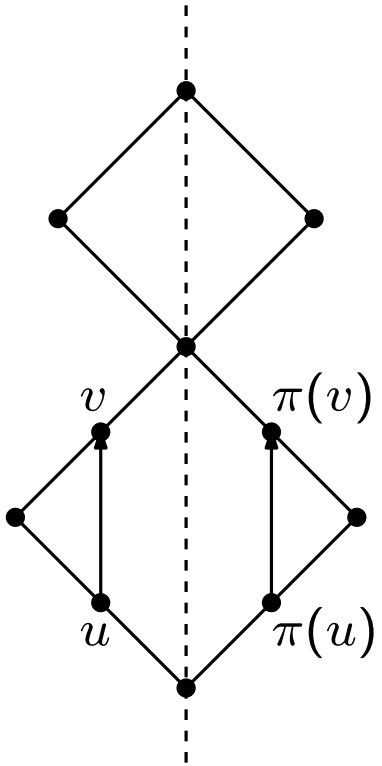
π_{vert} π_{hor}

Symmetrien in SP-Graphen

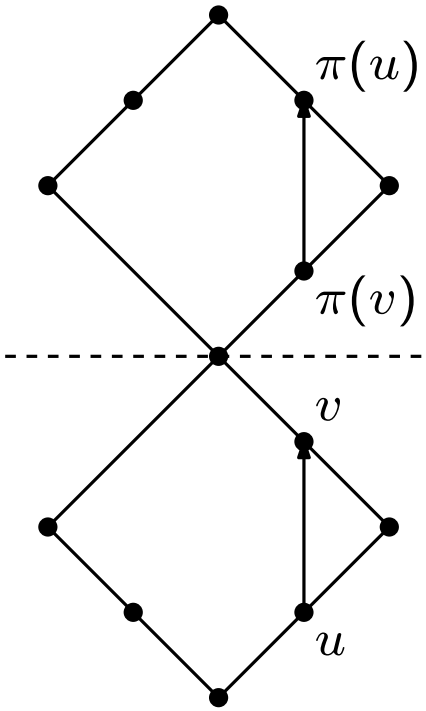


π_{vert} π_{hor} π_{rot}

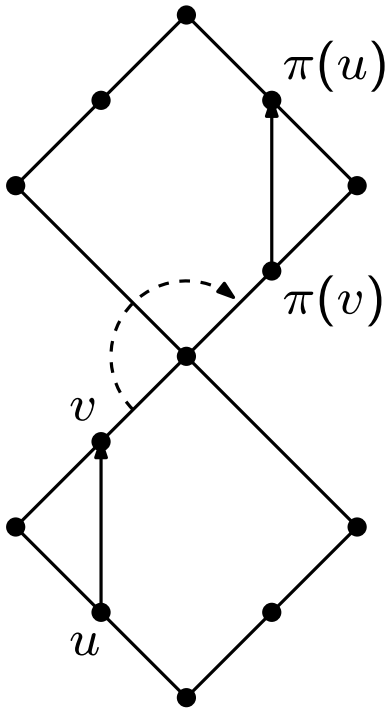
Symmetrien in SP-Graphen



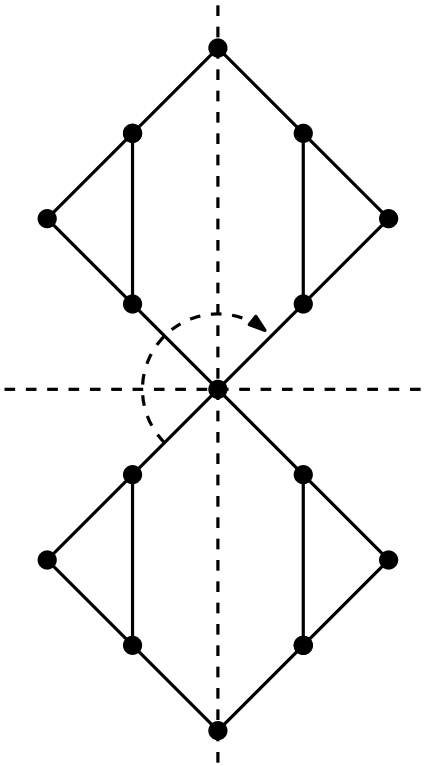
π_{vert}



π_{hor}



π_{rot}



$\{\pi_{\text{vert}}, \pi_{\text{hor}}, \pi_{\text{rot}}\}$

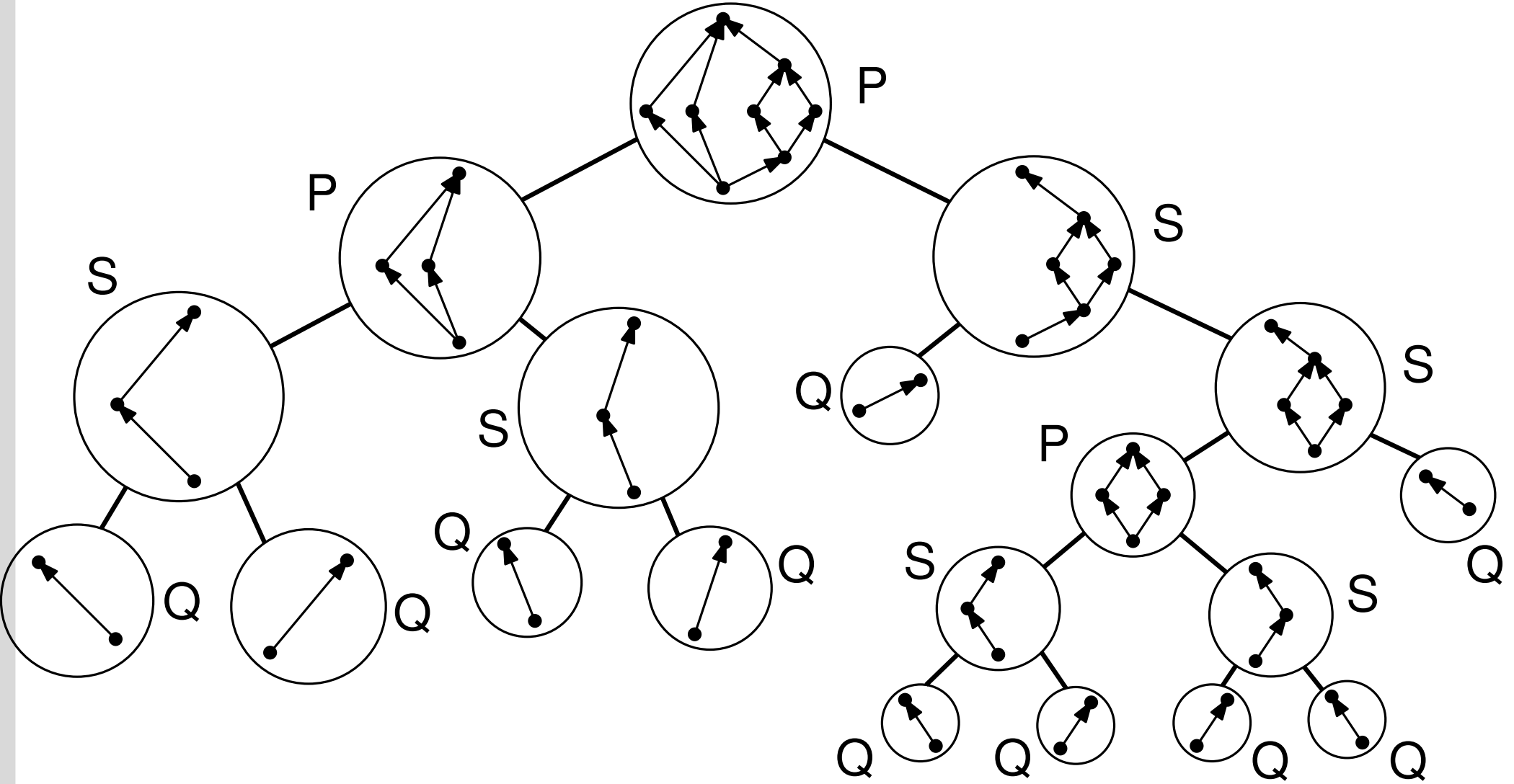
Symmetrien in SP-Zeichnungen

Satz (Hong, Eades, Lee '00)

Die in einem kreuzungsfreien Aufwärtslayout eines SP-Graphen darstellbaren Symmetrien sind entweder

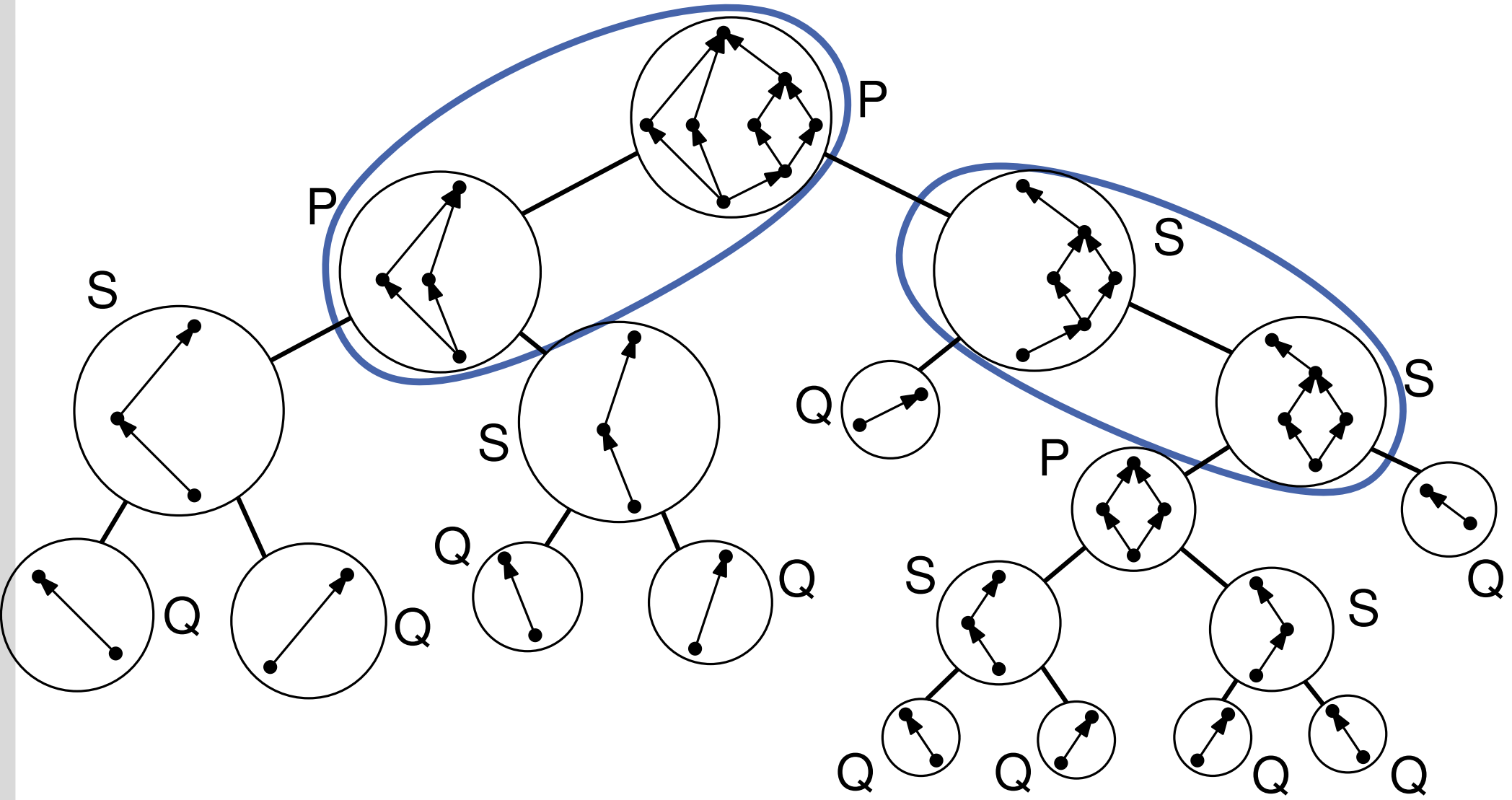
- $\{\mathrm{id}\}$
- $\{\mathrm{id}, \pi\}$ mit $\pi \in \{\pi_{\mathrm{vert}}, \pi_{\mathrm{hor}}, \pi_{\mathrm{rot}}\}$
- $\{\mathrm{id}, \pi_{\mathrm{vert}}, \pi_{\mathrm{hor}}, \pi_{\mathrm{rot}}\}$.

Knotenkodierung im Dekompositionsbaum

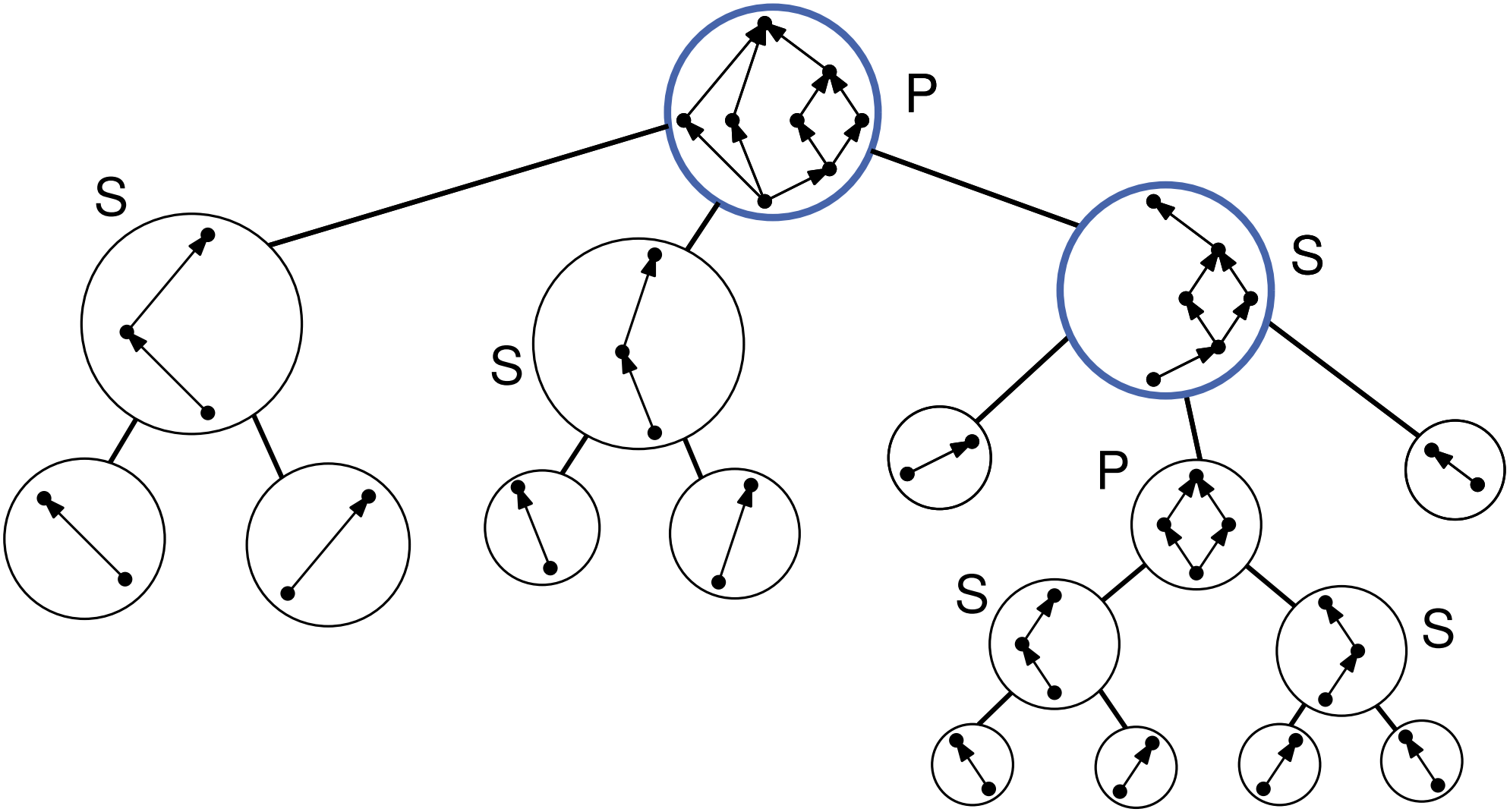


Institute for Theoretical Informatics
Algorithmics Group I

Knotenkodierung im Dekompositionsbaum



Knotenkodierung im Dekompositionsbaum



Institute for Theoretical Informatics
Algorithmics Group I

Knotenkodierung im Dekompositionsbaum

Kanonische Kodierung

- Setze $C(G) = \langle 0 \rangle$ für alle Q-Knoten G.

Knotenkodierung im Dekompositionsbaum

Kanonische Kodierung

- Setze $C(G) = \langle 0 \rangle$ für alle Q-Knoten G.
- Für jede Tiefe $t = \max_G \text{tiefe}(G), \ldots, 0$
  - Für jeden S- oder P-Knoten G der Tiefe t mit Nachfolgern $G_1, \ldots, G_k$ setze $C(G) = \langle c(G_1), \ldots, c(G_k) \rangle$ und sortiere $C(G)$ nichtabsteigend, falls G ein P-Knoten ist.

Knotenkodierung im Dekompositionsbaum

Kanonische Kodierung

- Setze $C(G) = \langle 0 \rangle$ für alle Q-Knoten G.
- Für jede Tiefe $t = \max_G \text{tiefe}(G), \ldots, 0$
  - Für jeden S- oder P-Knoten G der Tiefe t mit Nachfolgern $G_1, \ldots, G_k$ setze $C(G) = \langle c(G_1), \ldots, c(G_k) \rangle$ und sortiere $C(G)$ nichtabsteigend, falls G ein P-Knoten ist.
  - Sortiere die Menge der Tupel aller Knoten der Tiefe t lexikographisch.

Knotenkodierung im Dekompositionsbaum

Kanonische Kodierung

- Setze $C(G) = \langle 0 \rangle$ für alle Q-Knoten G.
- Für jede Tiefe $t = \max_G \text{tiefe}(G), \ldots, 0$
  - Für jeden S- oder P-Knoten G der Tiefe t mit Nachfolgern $G_1, \ldots, G_k$ setze $C(G) = \langle c(G_1), \ldots, c(G_k) \rangle$ und sortiere $C(G)$ nichtabsteigend, falls G ein P-Knoten ist.
  - Sortiere die Menge der Tupel aller Knoten der Tiefe t lexikographisch.
  - Für jede Komponente G der Tiefe t setze ihre Kodierung auf c, falls ihr Tupel in der sortierten Tupelfolge als c-tes verschiedenes Tupel auftritt.

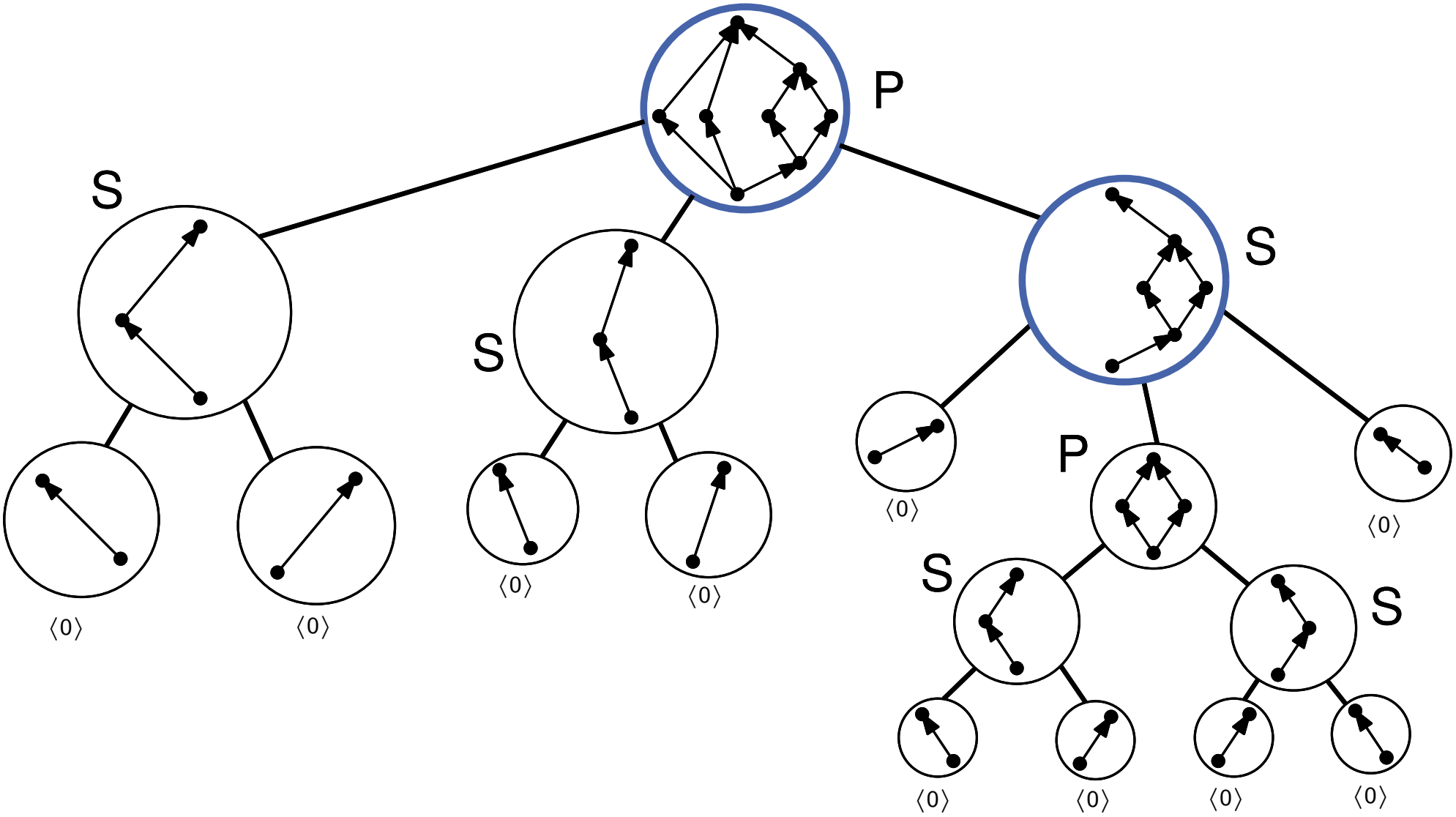
Knotenkodierung im Dekompositionsbaum

Kanonische Kodierung

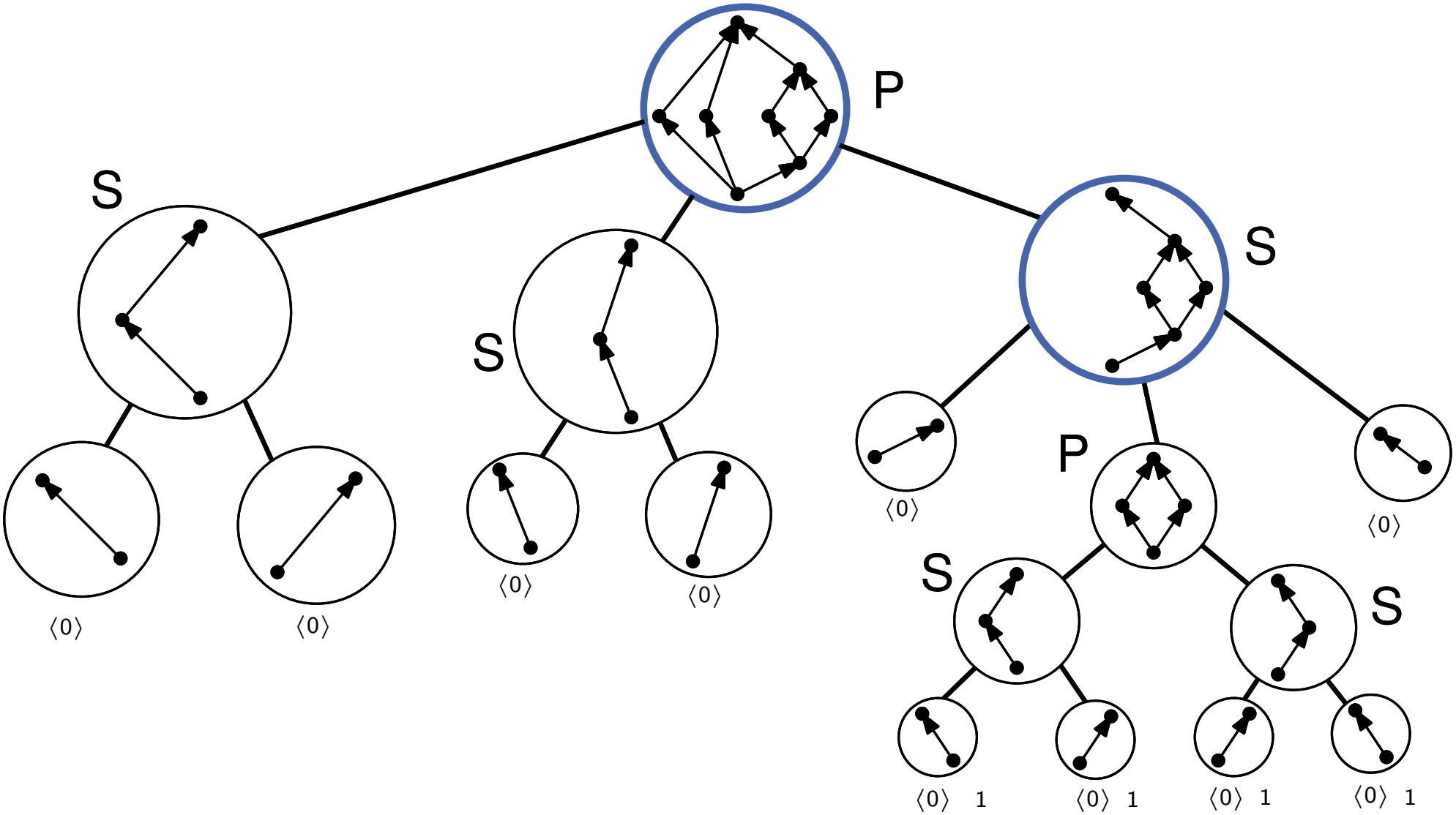
- Setze $C(G) = \langle 0 \rangle$ für alle Q-Knoten G.
- Für jede Tiefe $t = \max_G \text{tiefe}(G), \ldots, 0$
  - Für jeden S- oder P-Knoten G der Tiefe t mit Nachfolgern $G_1, \ldots, G_k$ setze $C(G) = \langle c(G_1), \ldots, c(G_k) \rangle$ und sortiere $C(G)$ nichtabsteigend, falls G ein P-Knoten ist.
  - Sortiere die Menge der Tupel aller Knoten der Tiefe t lexikographisch.
  - Für jede Komponente G der Tiefe t setze ihre Kodierung auf c, falls ihr Tupel in der sortierten Tupelfolge als c-tes verschiedenes Tupel auftritt.

Zwei Knoten u und v gleicher Tiefe sind genau dann isomorph, wenn sie den gleichen Code haben.

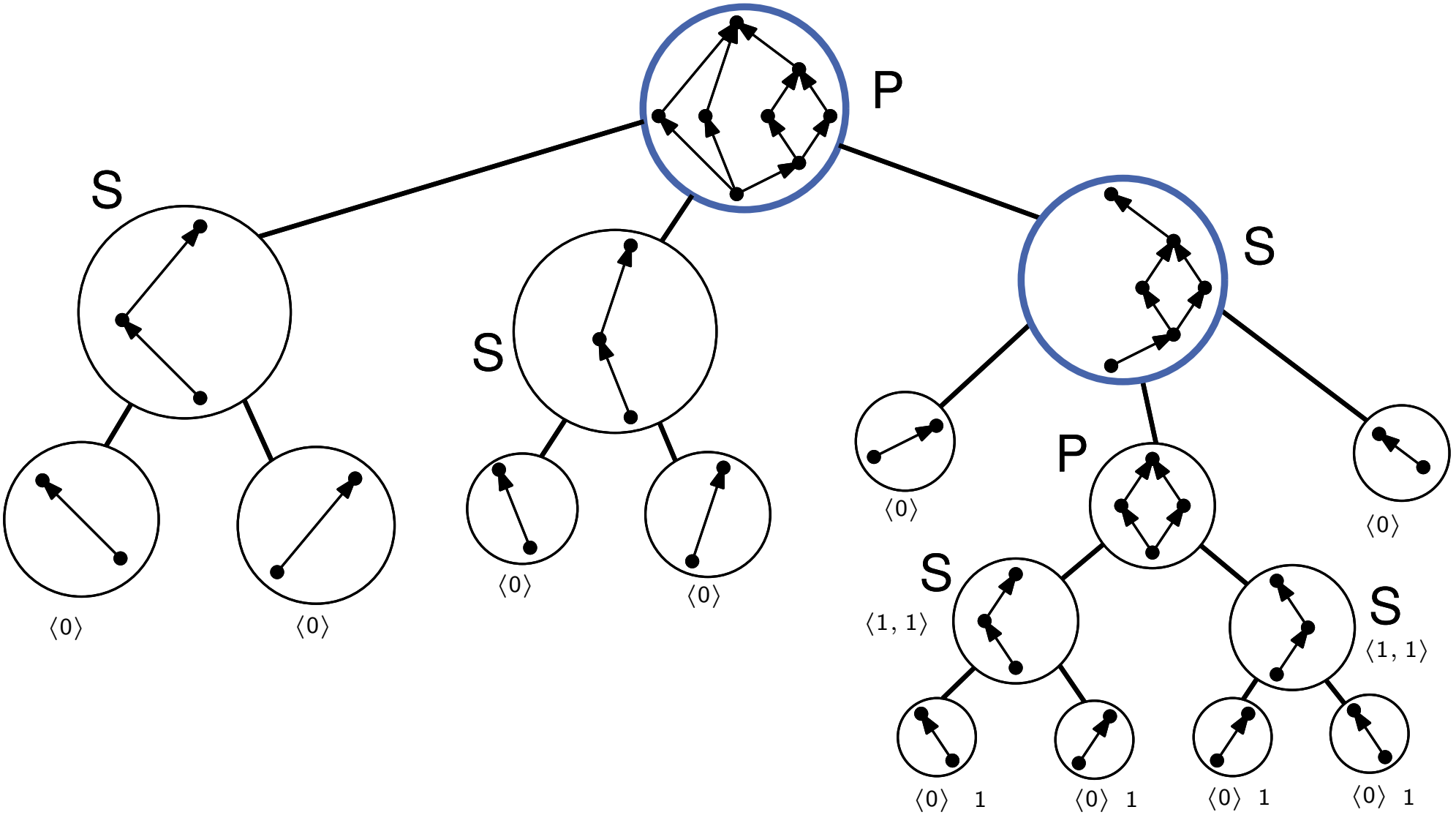
Knotenkodierung im Dekompositionsbaum



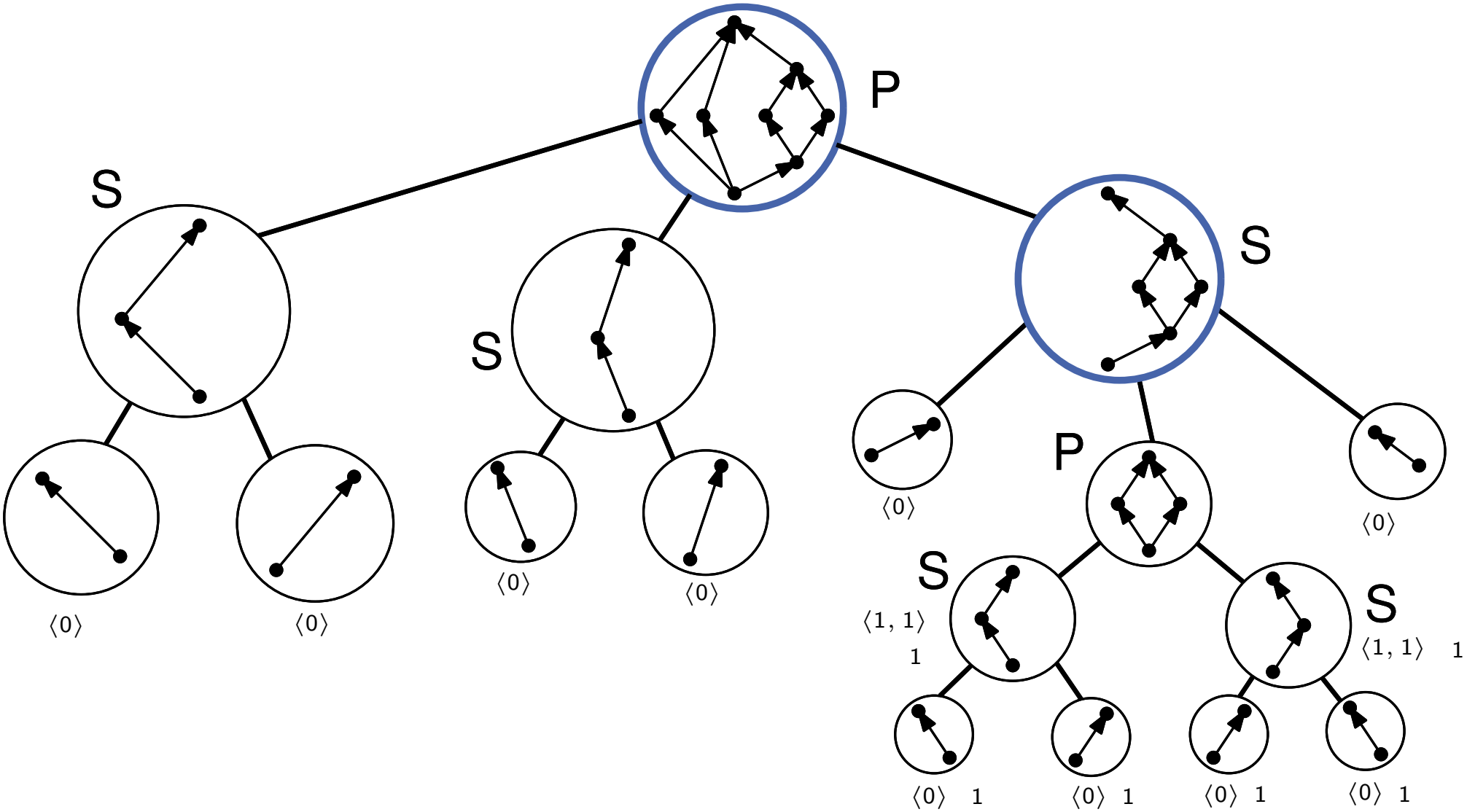
Knotenkodierung im Dekompositionsbaum



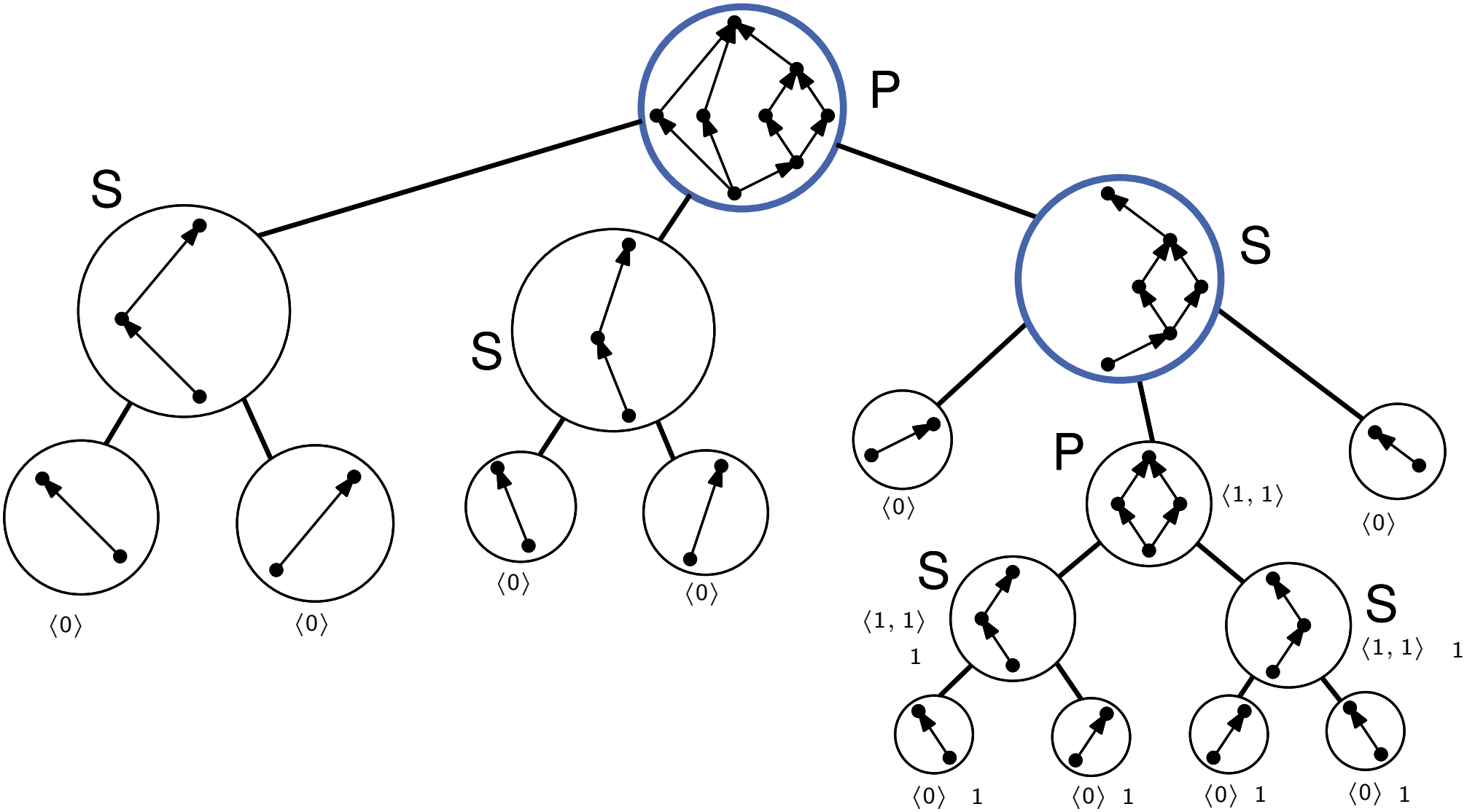
Knotenkodierung im Dekompositionsbaum



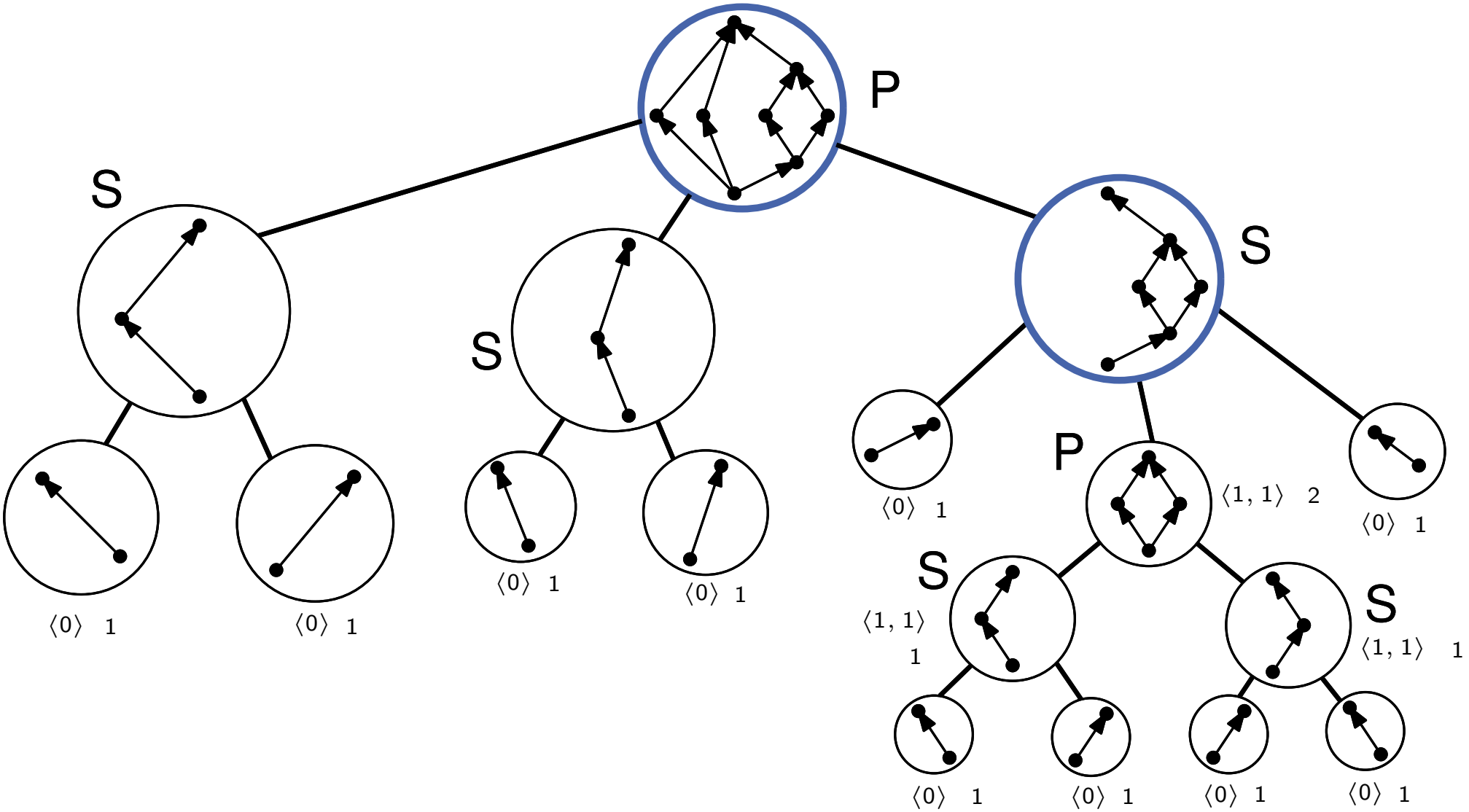
Knotenkodierung im Dekompositionsbaum



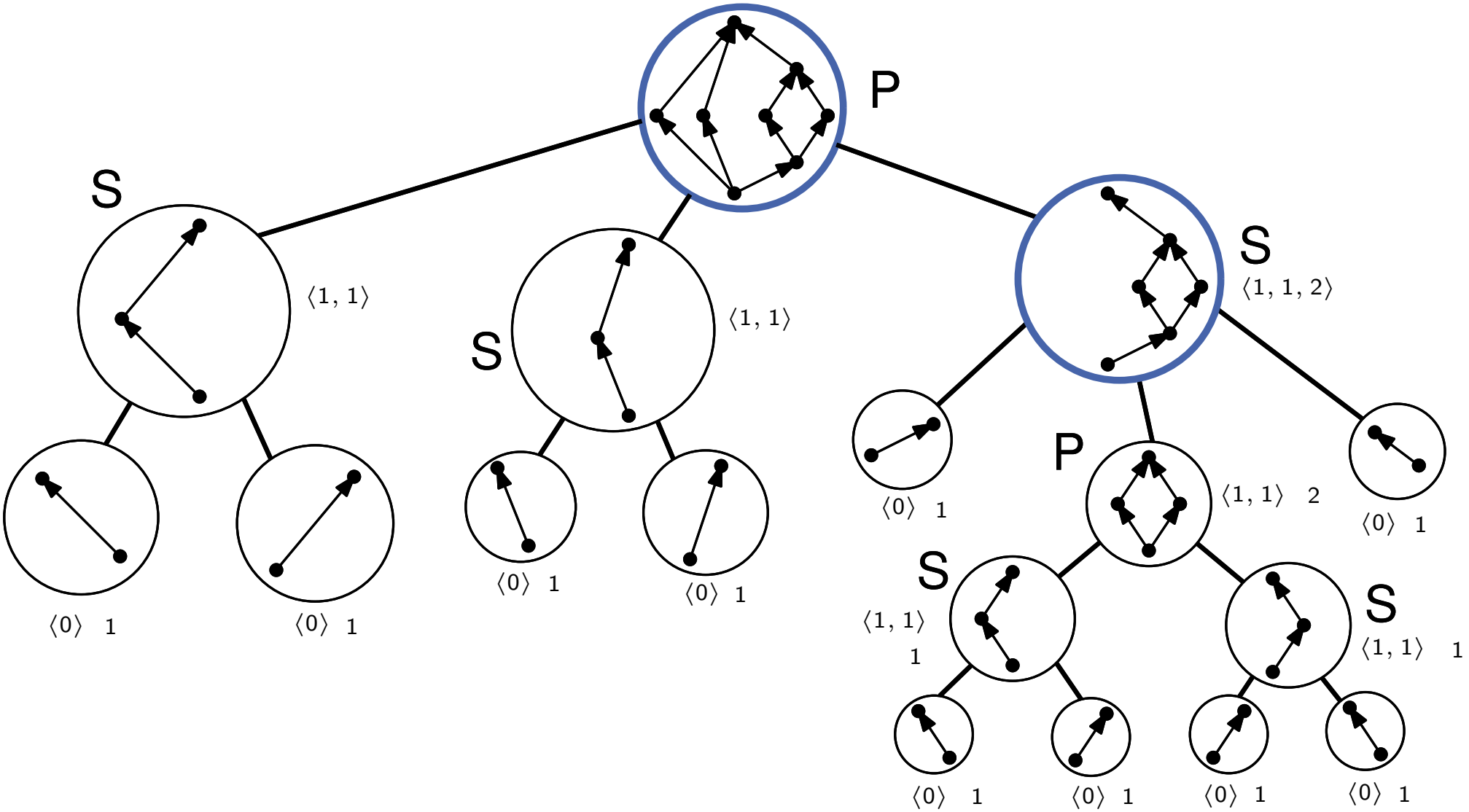
Knotenkodierung im Dekompositionsbaum



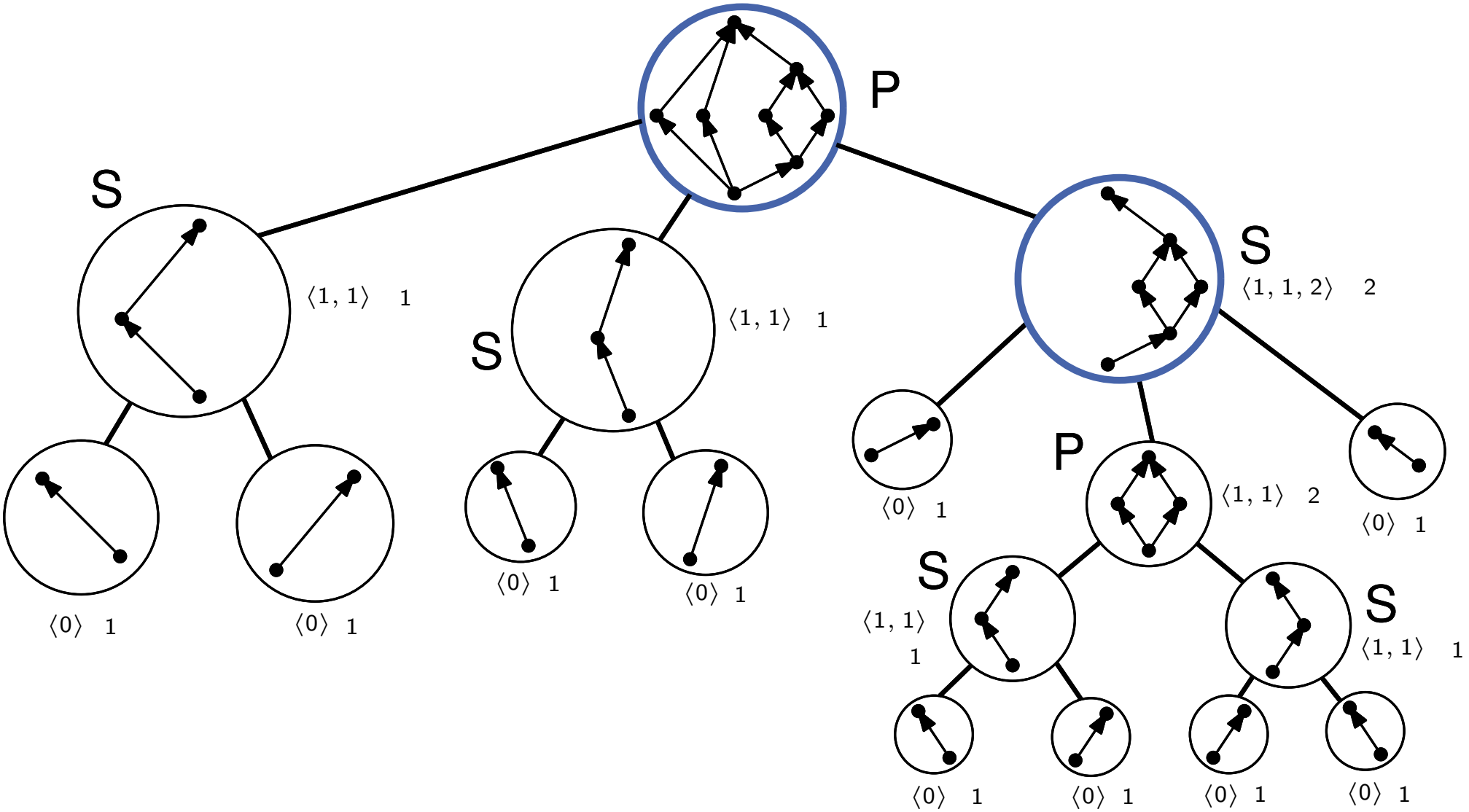
Knotenkodierung im Dekompositionsbaum



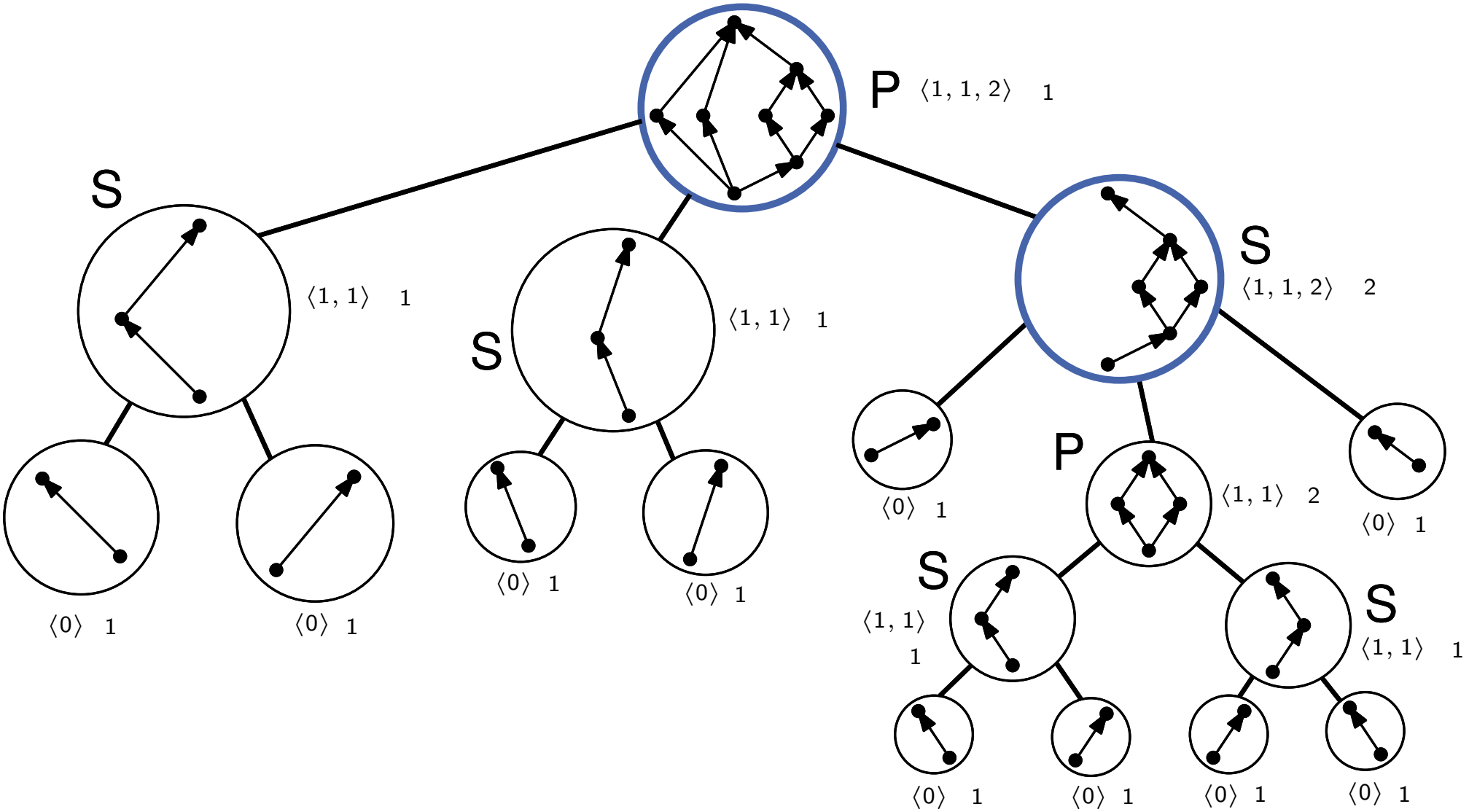
Knotenkodierung im Dekompositionsbaum



Knotenkodierung im Dekompositionsbaum



Knotenkodierung im Dekompositionsbaum



Vertikale Symmetrien

Satz (Hong, Eades, Lee '00)

Gegeben sei der kanonische Dekompositionsbaum eines serien-parallelen Graphen. Es sei G eine Komponente, die durch Komposition der Komponenten $G_1, \ldots, G_k$ entstehe.

- Ist G ein S-Knoten, dann ist G vertikal symmetrisch, wenn alle $G_1, \ldots, G_k$ vertikal symmetrisch sind.

Vertikale Symmetrien

Satz (Hong, Eades, Lee '00)

Gegeben sei der kanonische Dekompositionsbaum eines serien-parallelen Graphen. Es sei G eine Komponente, die durch Komposition der Komponenten $G_1, \ldots, G_k$ entstehe.

- Ist G ein S-Knoten, dann ist G vertikal symmetrisch, wenn alle $G_1, \ldots, G_k$ vertikal symmetrisch sind.
- Ist G ein P-Knoten, so betrachten wir die Klassen $\mathcal{G}_j = \{G_i : 1 \leq i \leq k, c(G_i) = j\}$, $j = 1, \ldots, k$, von isomorphen Teilgraphen.
  - $|\mathcal{G}_j|$ gerade $\forall j \Rightarrow$ vertikal symmetrisch

Vertikale Symmetrien

Satz (Hong, Eades, Lee '00)

Gegeben sei der kanonische Dekompositionsbaum eines serien-parallelen Graphen. Es sei G eine Komponente, die durch Komposition der Komponenten $G_1, \ldots, G_k$ entstehe.

- Ist G ein S-Knoten, dann ist G vertikal symmetrisch, wenn alle $G_1, \ldots, G_k$ vertikal symmetrisch sind.
- Ist G ein P-Knoten, so betrachten wir die Klassen $\mathcal{G}_j = \{G_i : 1 \leq i \leq k, c(G_i) = j\}$, $j = 1, \ldots, k$, von isomorphen Teilgraphen.
  - $|\mathcal{G}_j|$ gerade $\forall j \Rightarrow$ vertikal symmetrisch
  - $|\mathcal{G}_j|$ ungerade für genau ein $j \Rightarrow$ G vertikal symmetrisch g.d.w. Graphen in $\mathcal{G}_j$ vertikal symmetrisch

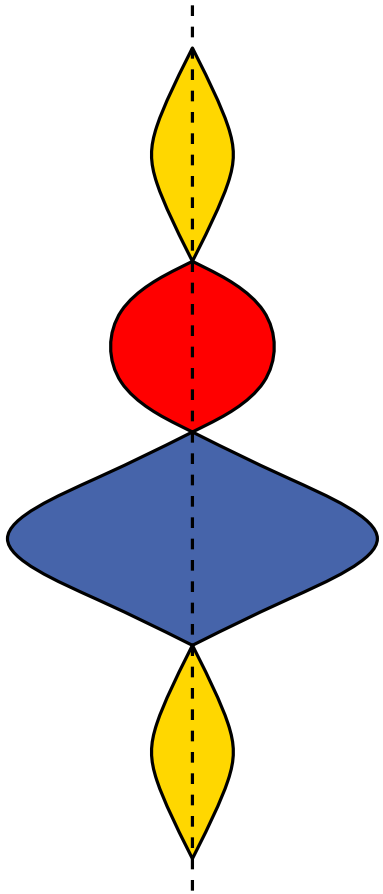
Vertikale Symmetrien

Satz (Hong, Eades, Lee '00)

Gegeben sei der kanonische Dekompositionsbaum eines serien-parallelen Graphen. Es sei G eine Komponente, die durch Komposition der Komponenten $G_1, \ldots, G_k$ entstehe.

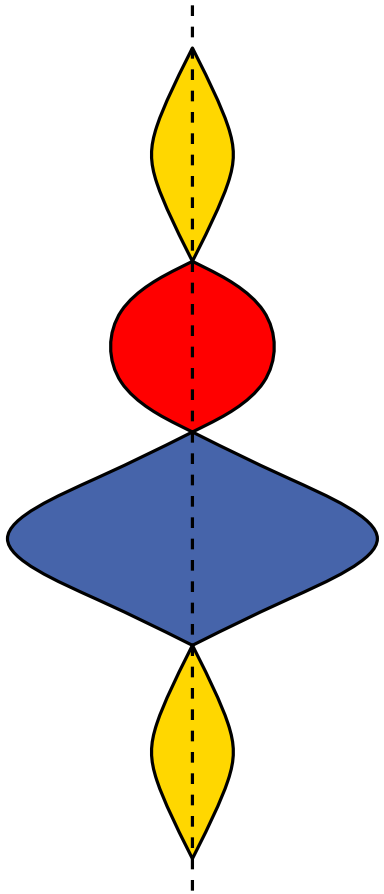
- Ist G ein S-Knoten, dann ist G vertikal symmetrisch, wenn alle $G_1, \ldots, G_k$ vertikal symmetrisch sind.
- Ist G ein P-Knoten, so betrachten wir die Klassen $\mathcal{G}_j = \{G_i : 1 \leq i \leq k, c(G_i) = j\}$, $j = 1, \ldots, k$, von isomorphen Teilgraphen.
  - $|\mathcal{G}_j|$ gerade $\forall j \Rightarrow$ vertikal symmetrisch
  - $|\mathcal{G}_j|$ ungerade für genau ein $j \Rightarrow$ G vertikal symmetrisch g.d.w. Graphen in $\mathcal{G}_j$ vertikal symmetrisch
  - $|\mathcal{G}_i|, |\mathcal{G}_j|$ ungerade für $i \neq j \Rightarrow G$ nicht vertikal symmetrisch

Beweisidee vertikale Symmetrie

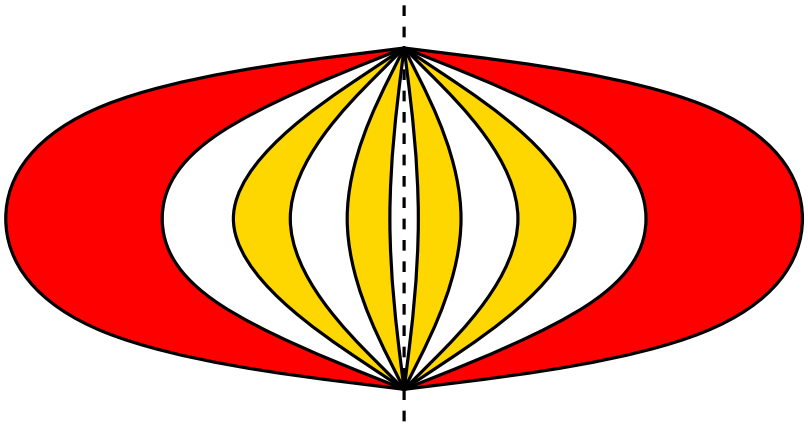


S-Knoten

Beweisidee vertikale Symmetrie

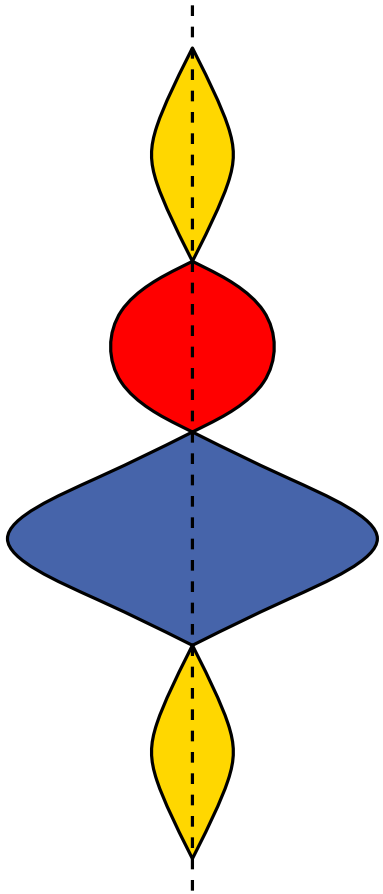


S-Knoten

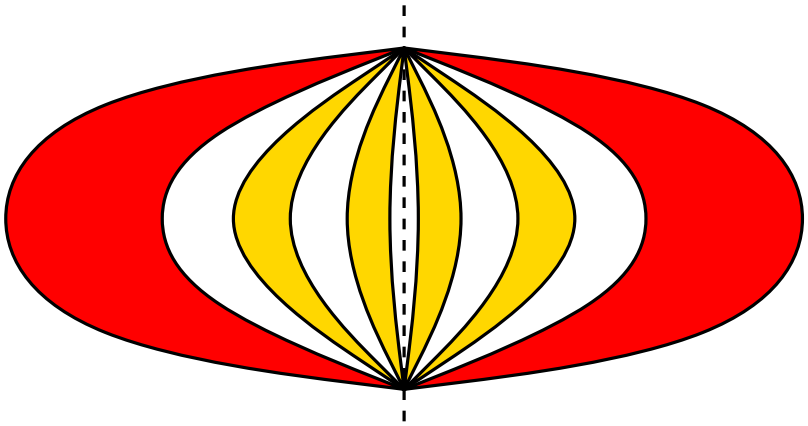


P-Knoten, alle Klassen gerade Anzahl

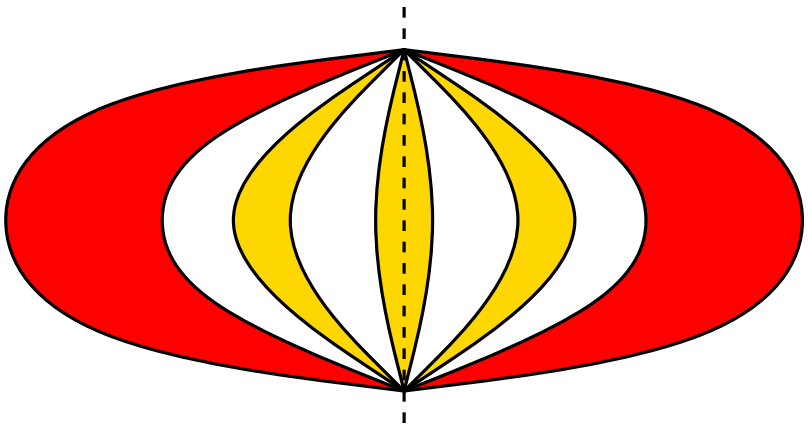
Beweisidee vertikale Symmetrie



S-Knoten
alle Knoten v-symm.

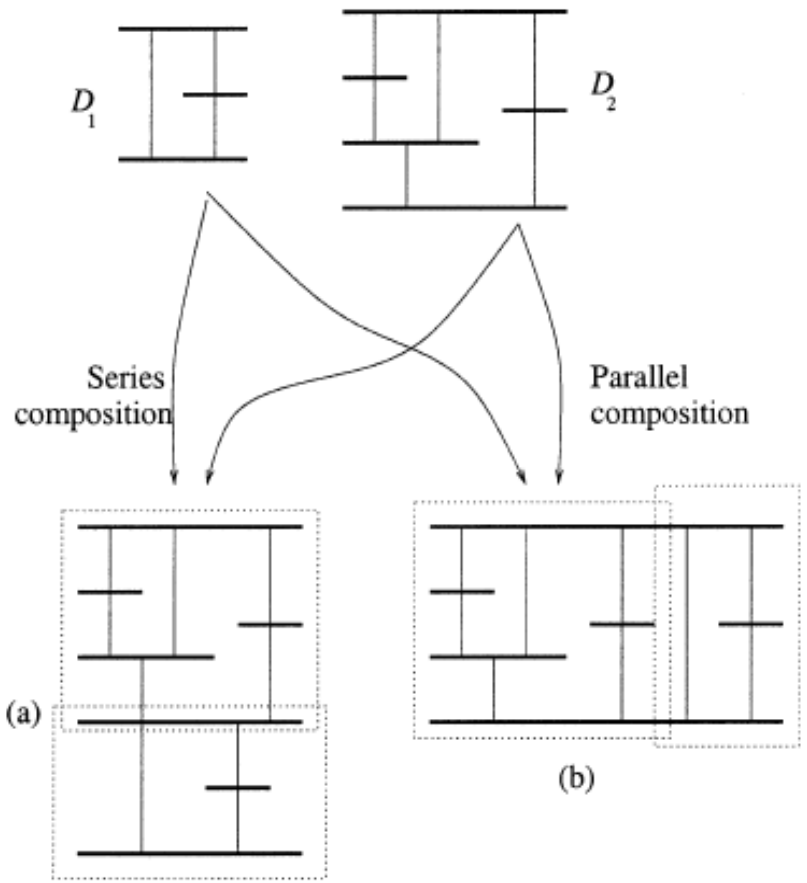


P-Knoten, alle Klassen gerade Anzahl



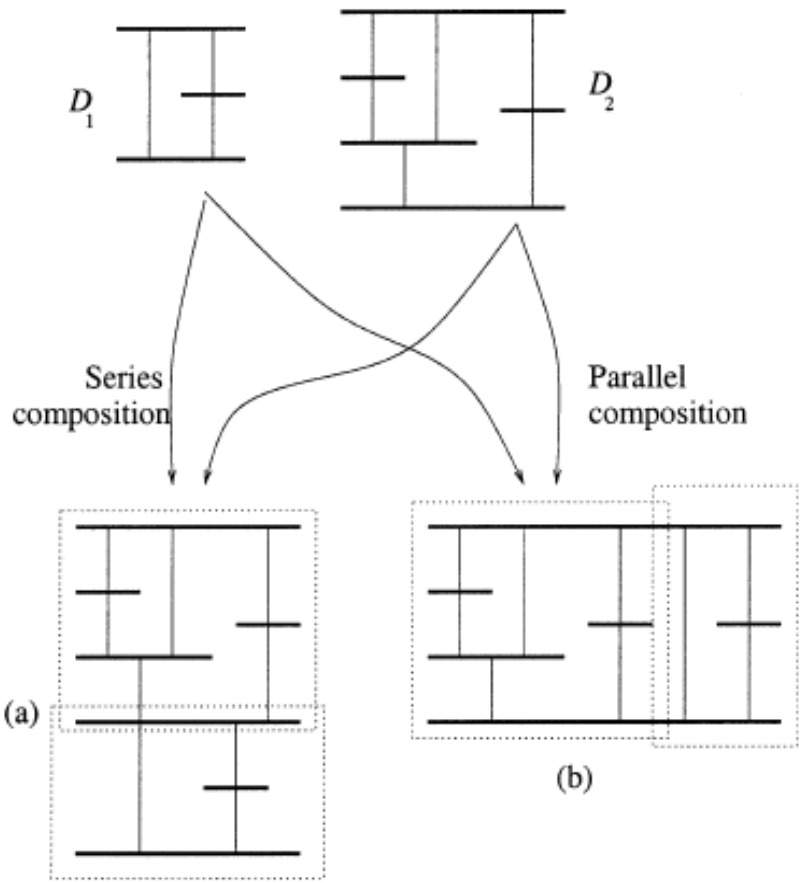
P-Knoten, eine Klasse ungerade Anzahl & v-symm.

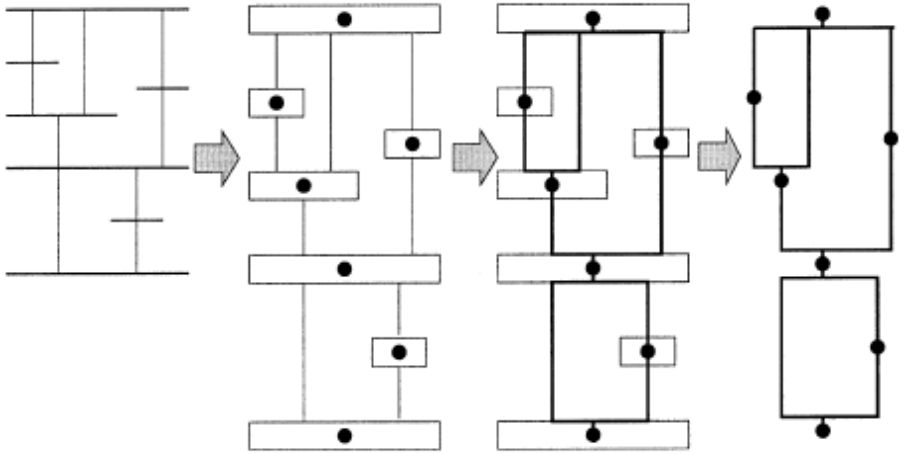
Symmetrien zeichnen



aus Hong, Eades, Lee ’00

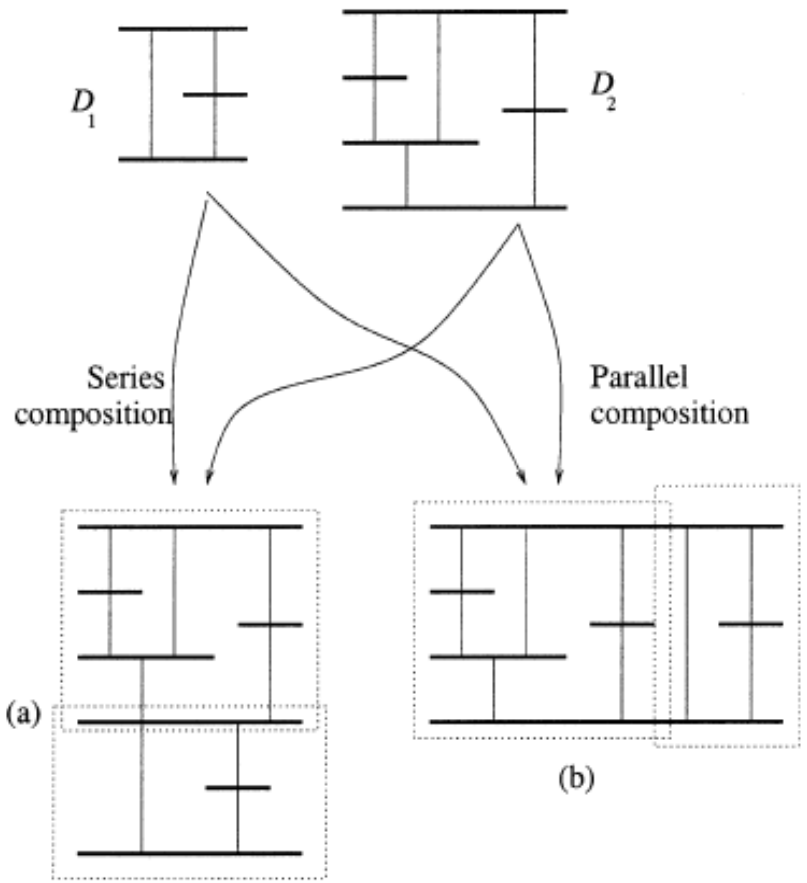
Symmetrien zeichnen

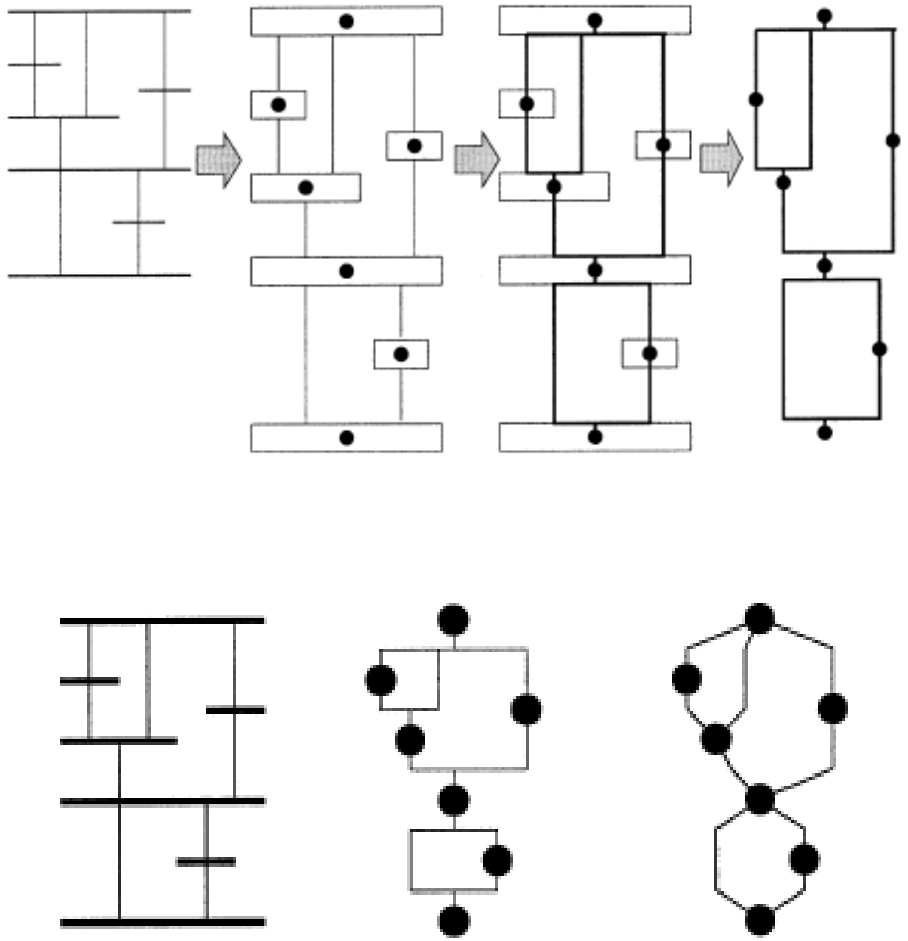




aus Hong, Eades, Lee ’00

Symmetrien zeichnen





aus Hong, Eades, Lee ’00