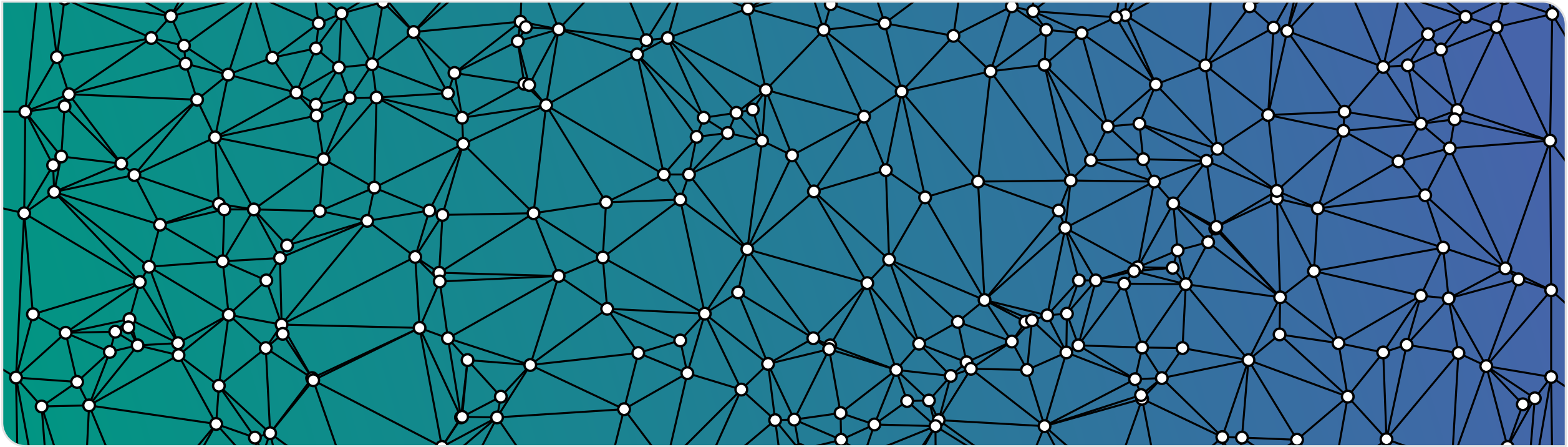
Karlsruher Institut für Technologie

Chains, Koch Chains, and Point Sets with Many Triangulations

von Daniel Rutschmann und Manuel Wettstein

Thomas Schäfer

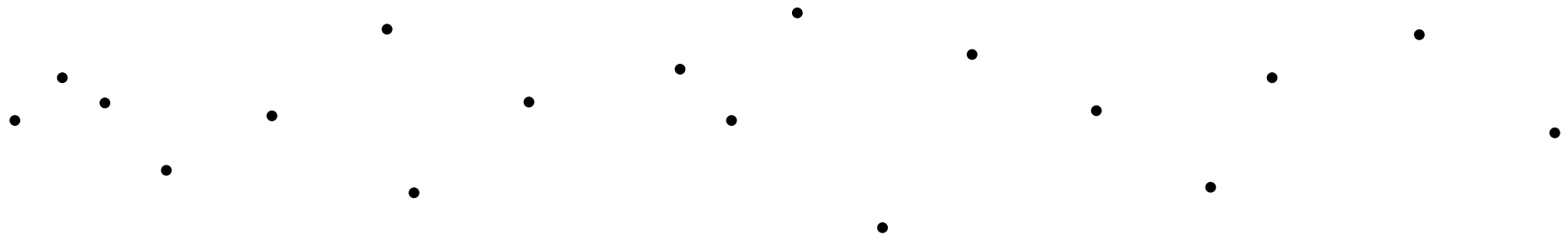


Triangulierung von Punktmengen

- Triangulierung eines der klassischen Probleme der algorithmischen Geometrie

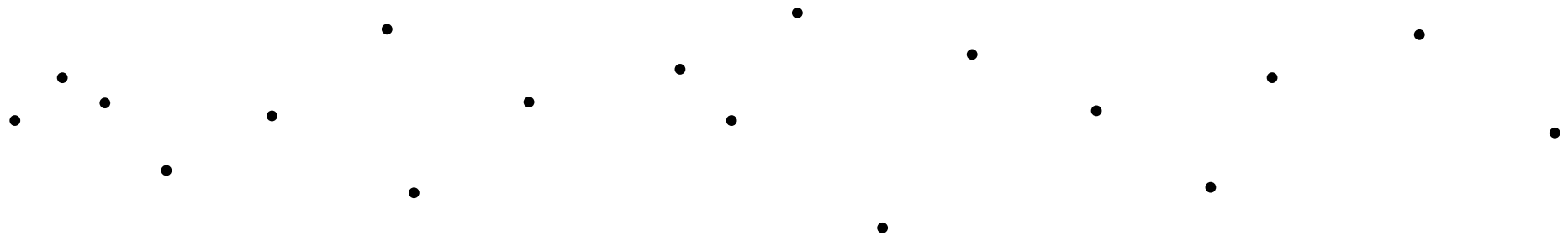
Triangulierung von Punktmengen

- Triangulierung eines der klassischen Probleme der algorithmischen Geometrie
- Gegeben: Punktmenge P von n Punkten in der euklidischen Ebene
  - Annahme: allgemeine Lage von P



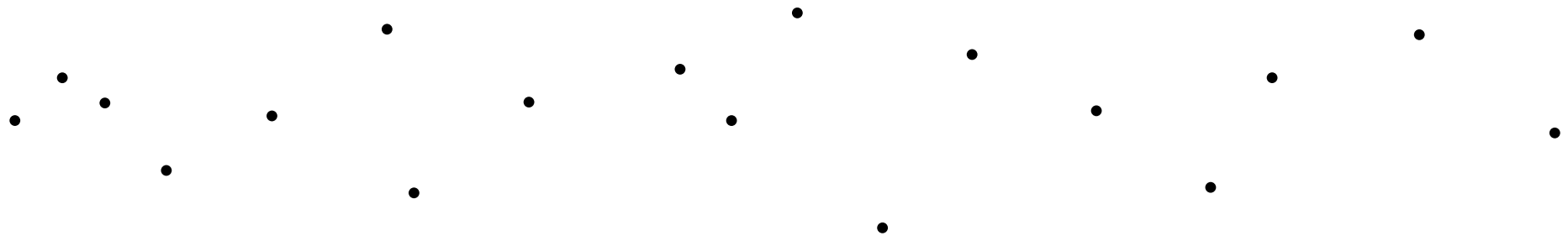
Triangulierung von Punktmengen

- Triangulierung eines der klassischen Probleme der algorithmischen Geometrie
- Gegeben: Punktmenge P von n Punkten in der euklidischen Ebene
  - Annahme: allgemeine Lage von P
- Triangulierung von Punktmenge ist ein kreuzungsfreier Graph mit nur geradlinigen Kanten auf P



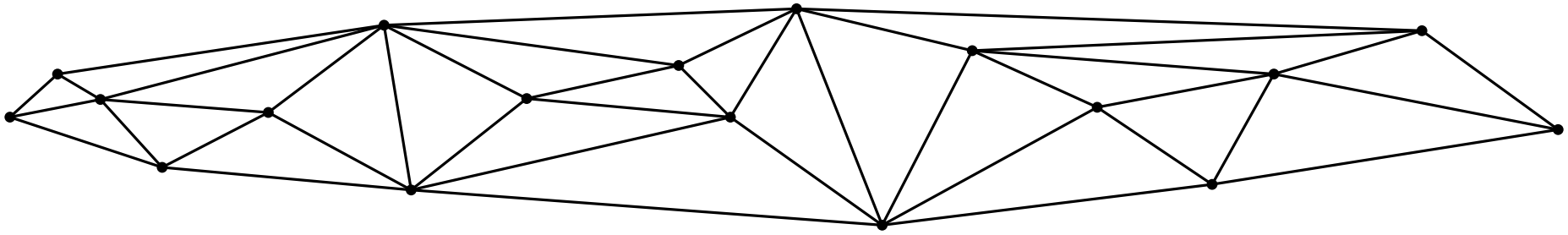
Triangulierung von Punktmengen

- Triangulierung eines der klassischen Probleme der algorithmischen Geometrie
- Gegeben: Punktmenge P von n Punkten in der euklidischen Ebene
  - Annahme: allgemeine Lage von P
- Triangulierung von Punktmenge ist ein kreuzungsfreier Graph mit nur geradlinigen Kanten auf P
- Wie viele verschiedene Triangulierungen gibt es für P?



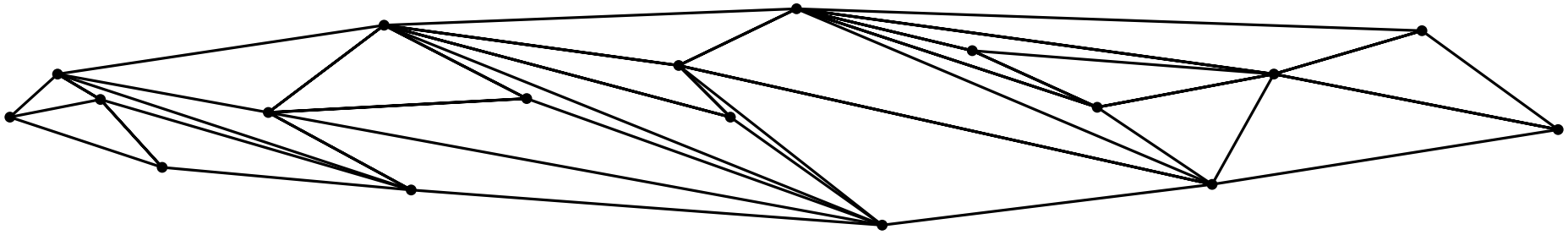
Triangulierung von Punktmengen

- Triangulierung eines der klassischen Probleme der algorithmischen Geometrie
- Gegeben: Punktmenge P von n Punkten in der euklidischen Ebene
  - Annahme: allgemeine Lage von P
- Triangulierung von Punktmenge ist ein kreuzungsfreier Graph mit nur geradlinigen Kanten auf P
- Wie viele verschiedene Triangulierungen gibt es für P?



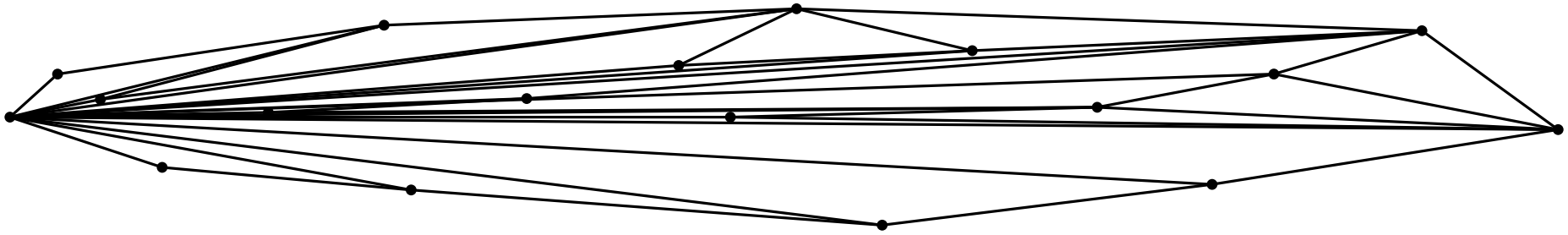
Triangulierung von Punktmengen

- Triangulierung eines der klassischen Probleme der algorithmischen Geometrie
- Gegeben: Punktmenge P von n Punkten in der euklidischen Ebene
  - Annahme: allgemeine Lage von P
- Triangulierung von Punktmenge ist ein kreuzungsfreier Graph mit nur geradlinigen Kanten auf P
- Wie viele verschiedene Triangulierungen gibt es für P?



Triangulierung von Punktmengen

- Triangulierung eines der klassischen Probleme der algorithmischen Geometrie
- Gegeben: Punktmenge P von n Punkten in der euklidischen Ebene
  - Annahme: allgemeine Lage von P
- Triangulierung von Punktmenge ist ein kreuzungsfreier Graph mit nur geradlinigen Kanten auf P
- Wie viele verschiedene Triangulierungen gibt es für P?



Familien von Punktmengen

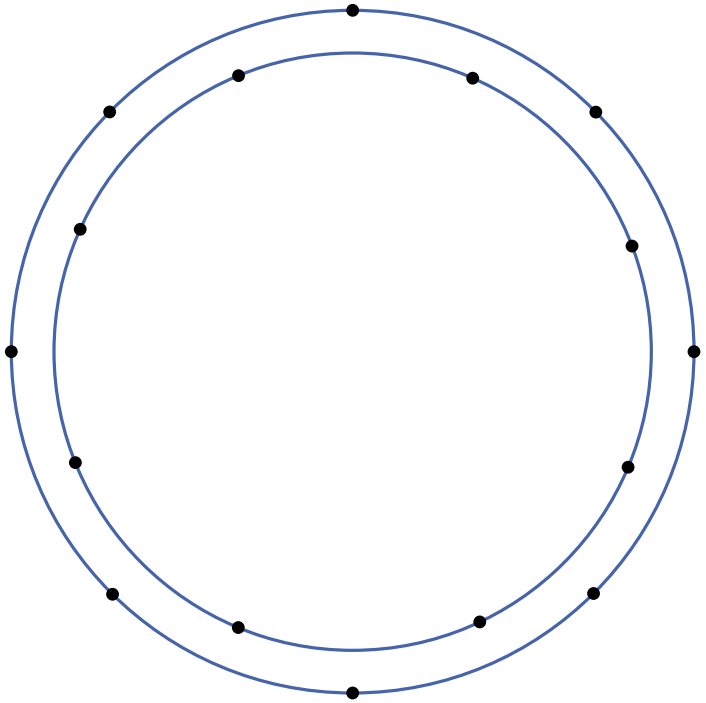
- gesucht: Familie von Punktmengen mit besonders vielen Triangulierungen

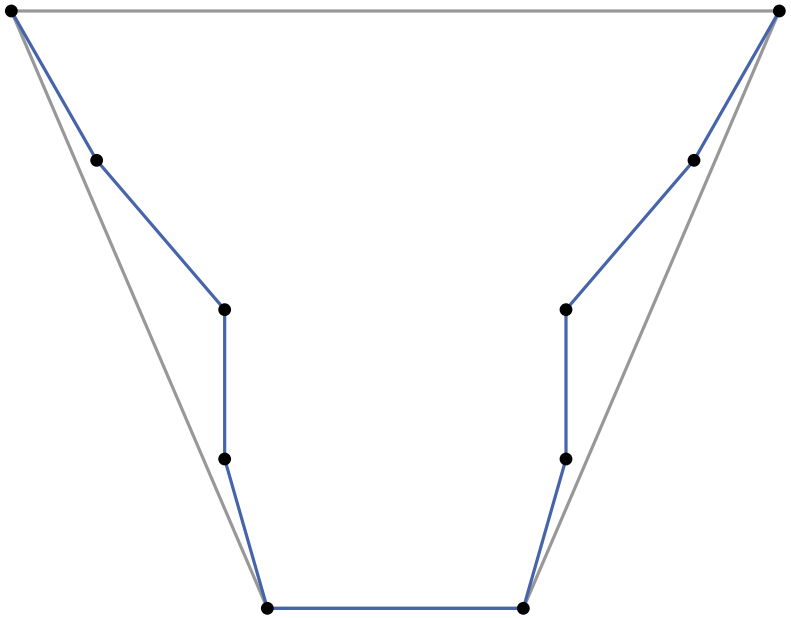
Familien von Punktmengen

- gesucht: Familie von Punktmengen mit besonders vielen Triangulierungen
- Bezeichne tr(P) als Anzahl verschiedener Triangulierungen einer gegebenen Punktemenge P

Familien von Punktmengen

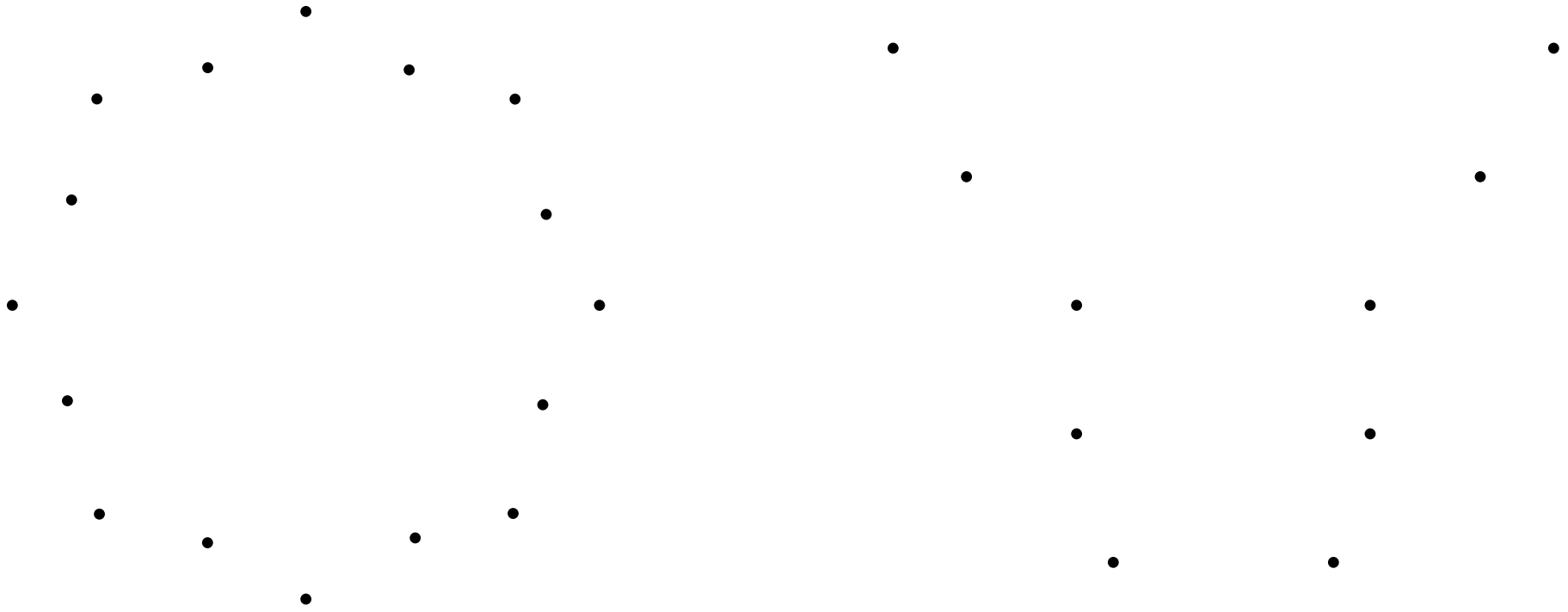
- gesucht: Familie von Punktmengen mit besonders vielen Triangulierungen
- Bezeichne tr(P) als Anzahl verschiedener Triangulierungen einer gegebenen Punktemenge P





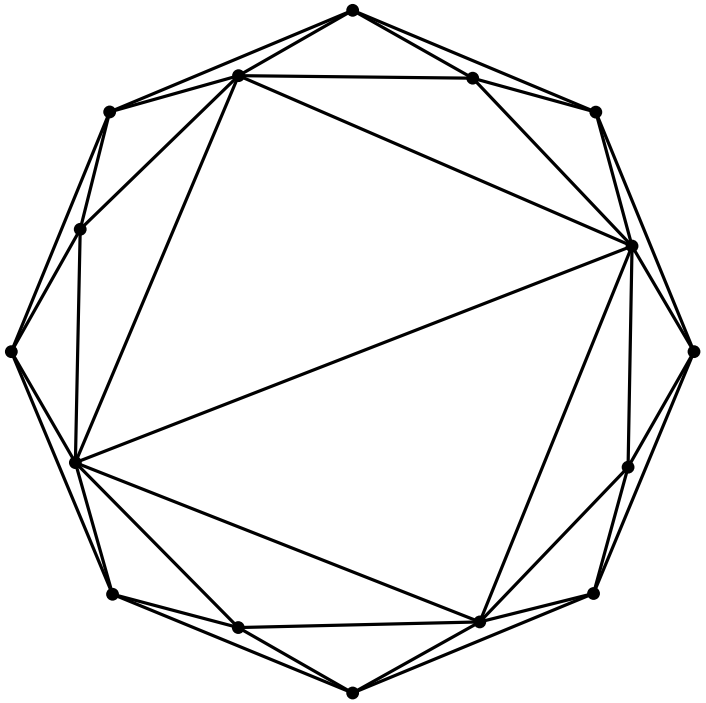
Familien von Punktmengen

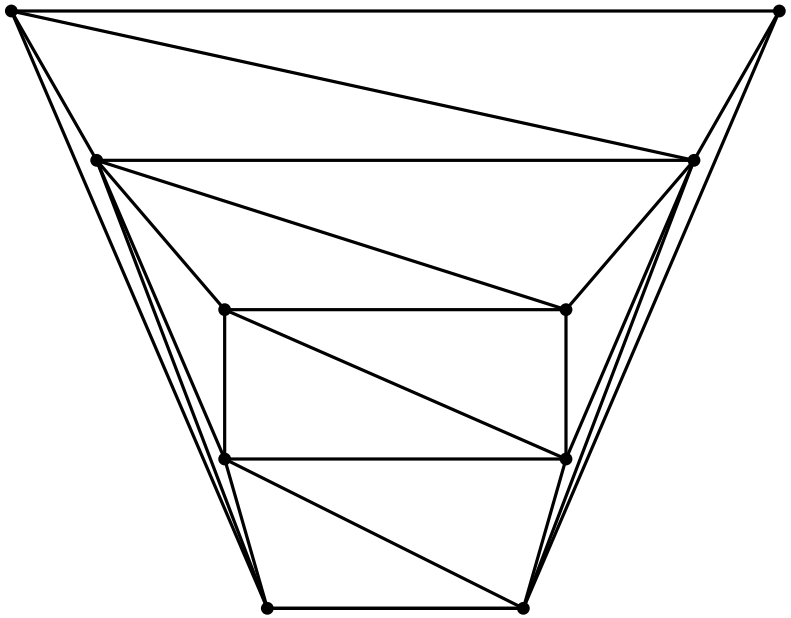
- gesucht: Familie von Punktmengen mit besonders vielen Triangulierungen
- Bezeichne tr(P) als Anzahl verschiedener Triangulierungen einer gegebenen Punktemenge P



Familien von Punktmengen

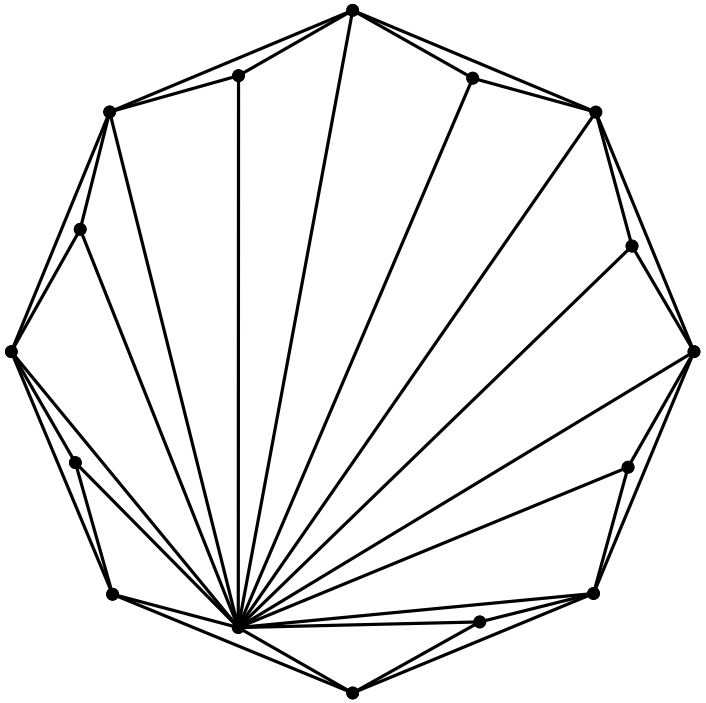
- gesucht: Familie von Punktmengen mit besonders vielen Triangulierungen
- Bezeichne tr(P) als Anzahl verschiedener Triangulierungen einer gegebenen Punktemenge P



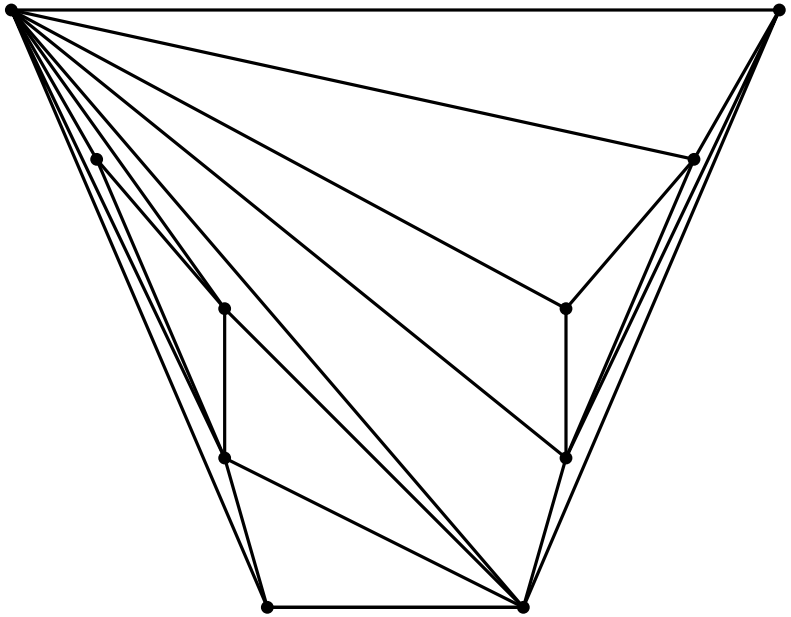


Familien von Punktmengen

- gesucht: Familie von Punktmengen mit besonders vielen Triangulierungen
- Bezeichne $\mathrm{tr}(P)$ als Anzahl verschiedener Triangulierungen einer gegebenen Punktemenge P



$\mathrm{tr}(P) = \Theta(3.47^n)$ [Dumitrescu et al. 2013]



$\mathrm{tr}(P) = \Omega(8.65^n)$ [Hurtado, Noy 1997]

Contribution des Papers

- neue Techniken zur abstrakten Notation von Punktmengen
- lassen sich dadurch als Formel von Operationen darstellen

Contribution des Papers

- neue Techniken zur abstrakten Notation von Punktmengen
- lassen sich dadurch als Formel von Operationen darstellen
- Kombinatorik statt Geometrie
- ermöglicht kombinatorischen Beweis der Schranken

Contribution des Papers

- neue Techniken zur abstrakten Notation von Punktmengen
- lassen sich dadurch als Formel von Operationen darstellen
- Kombinatorik statt Geometrie
- ermöglicht kombinatorischen Beweis der Schranken
- Vorstellung neue Familie von Punktmengen mit bisher meisten verschiedenen Triangulierungen (Koch Chains)

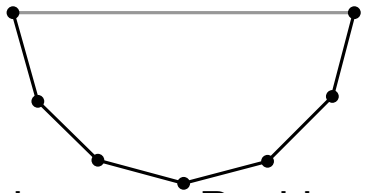
Chains

- letzte Rekordhalter lassen sich alle Typ *Chain* zuordnen
- Chain C ist Sequenz von Punkten $p_0, \ldots, p_n$ sortiert nach x-Koordinate, sodass jede Kante $p_{i-1}p_i$ *unvermeidbar* in Triangulierung von C

konvexe Position

Chains

- letzte Rekordhalter lassen sich alle Typ *Chain* zuordnen
- Chain C ist Sequenz von Punkten $p_0, \ldots, p_n$ sortiert nach x-Koordinate, sodass jede Kante $p_{i-1}p_i$ *unvermeidbar* in Triangulierung von C
- unvermeidbare Kanten auch als *Chain Edges* bezeichnet



konvexe Position

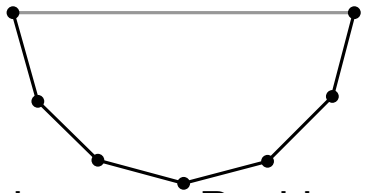
Chains

- letzte Rekordhalter lassen sich alle Typ *Chain* zuordnen
- Chain C ist Sequenz von Punkten $p_0, \ldots, p_n$ sortiert nach x-Koordinate, sodass jede Kante $p_{i-1}p_i$ *unvermeidbar* in Triangulierung von C
- unvermeidbare Kanten auch als *Chain Edges* bezeichnet
- Kante p_0p_n immer Kante der konvexen Hülle - auch unvermeidbar

konvexe Position

Chains

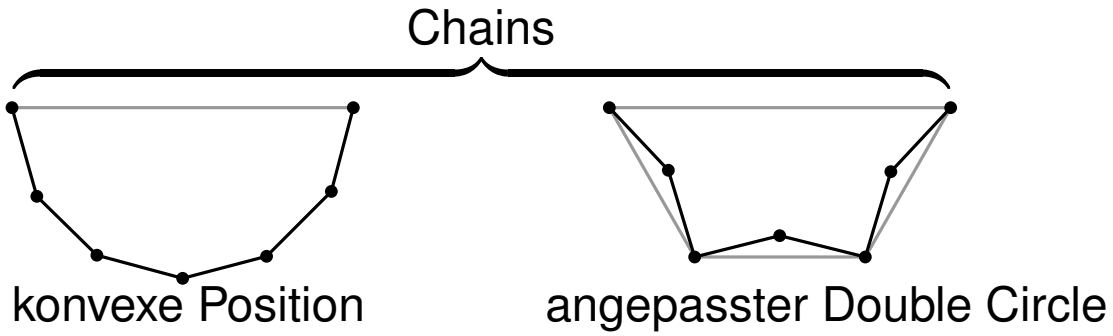
- letzte Rekordhalter lassen sich alle Typ *Chain* zuordnen
- Chain C ist Sequenz von Punkten $p_0, \ldots, p_n$ sortiert nach x-Koordinate, sodass jede Kante $p_{i-1}p_i$ *unvermeidbar* in Triangulierung von C
- unvermeidbare Kanten auch als *Chain Edges* bezeichnet
- Kante p_0p_n immer Kante der konvexen Hülle - auch unvermeidbar
- Chain Edges formen x-monotone *Chain Curve*



konvexe Position

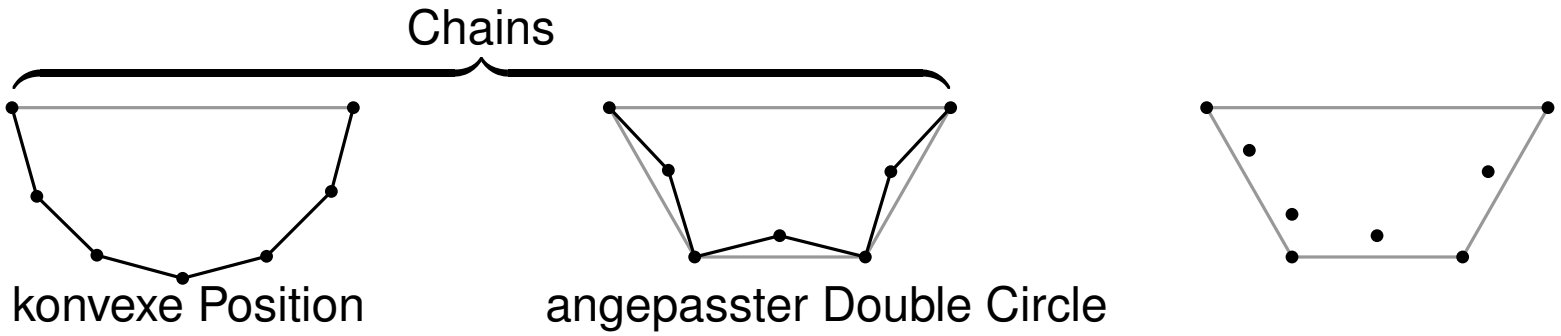
Chains

- letzte Rekordhalter lassen sich alle Typ *Chain* zuordnen
- Chain C ist Sequenz von Punkten $p_0, \ldots, p_n$ sortiert nach x-Koordinate, sodass jede Kante $p_{i-1}p_i$ *unvermeidbar* in Triangulierung von C
- unvermeidbare Kanten auch als *Chain Edges* bezeichnet
- Kante p_0p_n immer Kante der konvexen Hülle - auch unvermeidbar
- Chain Edges formen x-monotone *Chain Curve*



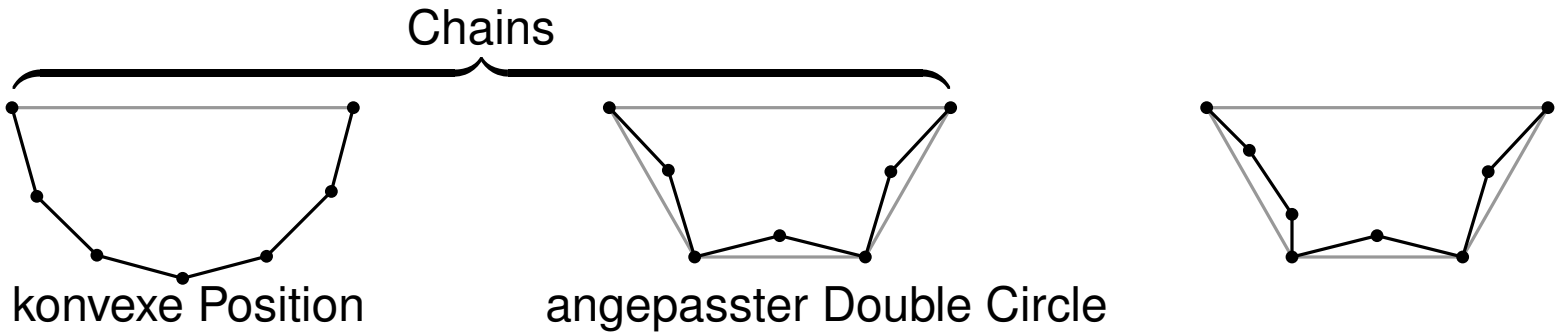
Chains

- letzte Rekordhalter lassen sich alle Typ *Chain* zuordnen
- Chain C ist Sequenz von Punkten $p_0, \ldots, p_n$ sortiert nach x-Koordinate, sodass jede Kante $p_{i-1}p_i$ *unvermeidbar* in Triangulierung von C
- unvermeidbare Kanten auch als *Chain Edges* bezeichnet
- Kante p_0p_n immer Kante der konvexen Hülle - auch unvermeidbar
- Chain Edges formen x-monotone *Chain Curve*



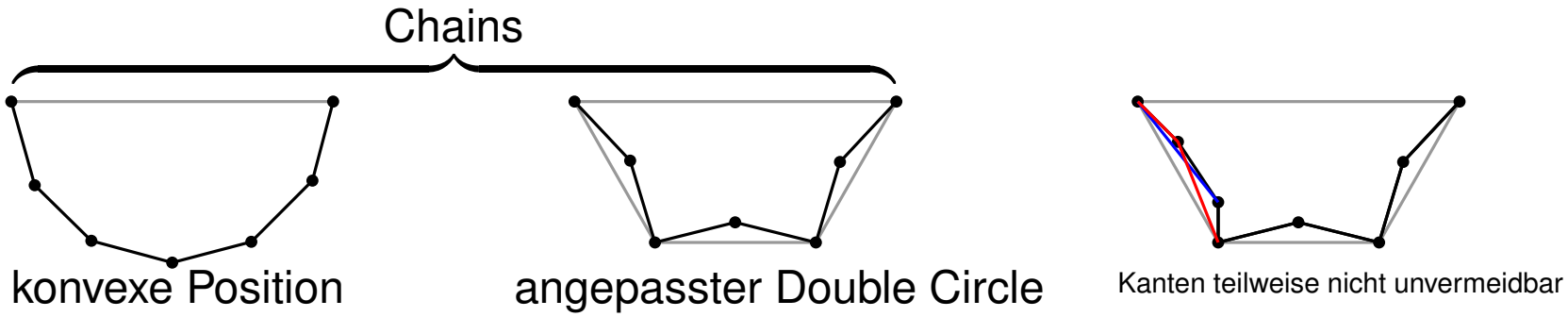
Chains

- letzte Rekordhalter lassen sich alle Typ *Chain* zuordnen
- Chain C ist Sequenz von Punkten $p_0, \ldots, p_n$ sortiert nach x-Koordinate, sodass jede Kante $p_{i-1}p_i$ *unvermeidbar* in Triangulierung von C
- unvermeidbare Kanten auch als *Chain Edges* bezeichnet
- Kante p_0p_n immer Kante der konvexen Hülle - auch unvermeidbar
- Chain Edges formen x-monotone *Chain Curve*



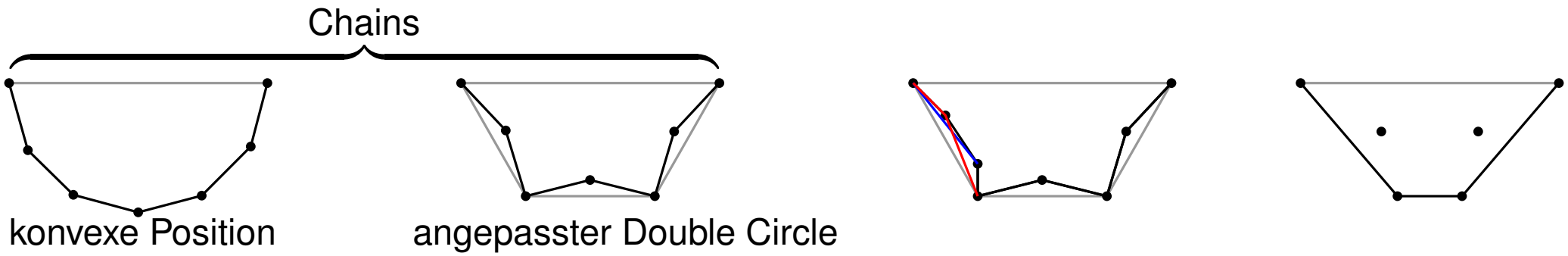
Chains

- letzte Rekordhalter lassen sich alle Typ *Chain* zuordnen
- Chain C ist Sequenz von Punkten $p_0, \ldots, p_n$ sortiert nach x-Koordinate, sodass jede Kante $p_{i-1}p_i$ *unvermeidbar* in Triangulierung von C
- unvermeidbare Kanten auch als *Chain Edges* bezeichnet
- Kante p_0p_n immer Kante der konvexen Hülle - auch unvermeidbar
- Chain Edges formen x-monotone *Chain Curve*



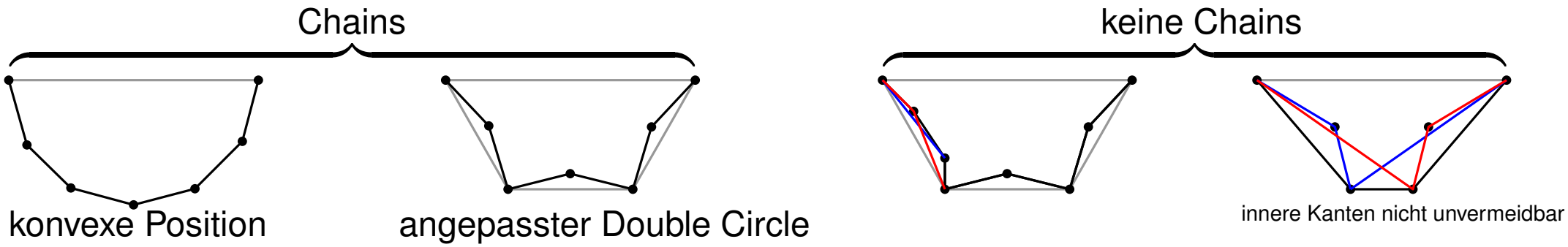
Chains

- letzte Rekordhalter lassen sich alle Typ *Chain* zuordnen
- Chain C ist Sequenz von Punkten $p_0, \ldots, p_n$ sortiert nach x-Koordinate, sodass jede Kante $p_{i-1}p_i$ *unvermeidbar* in Triangulierung von C
- unvermeidbare Kanten auch als *Chain Edges* bezeichnet
- Kante p_0p_n immer Kante der konvexen Hülle - auch unvermeidbar
- Chain Edges formen x-monotone *Chain Curve*



Chains

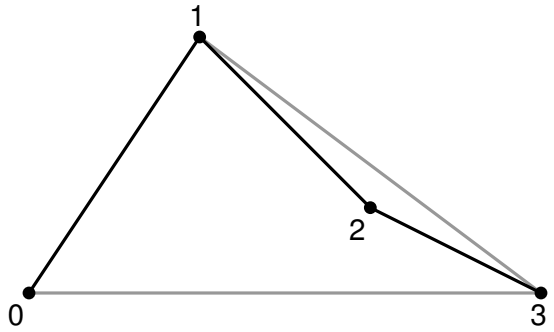
- letzte Rekordhalter lassen sich alle Typ *Chain* zuordnen
- Chain C ist Sequenz von Punkten $p_0, \ldots, p_n$ sortiert nach x-Koordinate, sodass jede Kante $p_{i-1}p_i$ *unvermeidbar* in Triangulierung von C
- unvermeidbare Kanten auch als *Chain Edges* bezeichnet
- Kante p_0p_n immer Kante der konvexen Hülle - auch unvermeidbar
- Chain Edges formen x-monotone *Chain Curve*



Strukturelle Eigenschaften von Chains - Visibility Triangle

- zu jeder Chain gibt es ein *Visibility Triangle* $V(C)$:

$$V(C)_{i,j} = \begin{cases} +1, & \text{falls } p_i p_j \text{ über Chain Curve (}\textit{obere Kante}\text{)}, \\ -1, & \text{falls } p_i p_j \text{ unter Chain Curve (}\textit{untere Kante}\text{)}, \\ 0, & \text{falls } p_i p_j \text{ Chain Edge.} \end{cases}$$

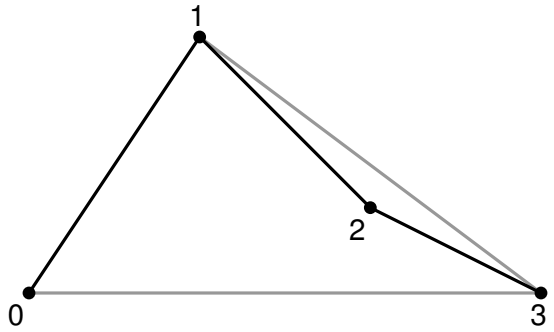


	1	2	3
0	0	-	-
1		0	+
2			0

Strukturelle Eigenschaften von Chains - Visibility Triangle

- zu jeder Chain gibt es ein *Visibility Triangle* $V(C)$:

$$V(C)_{i,j} = \begin{cases} +1, & \text{falls } p_i p_j \text{ über Chain Curve (}\textit{obere Kante}\text{)}, \\ -1, & \text{falls } p_i p_j \text{ unter Chain Curve (}\textit{untere Kante}\text{)}, \\ 0, & \text{falls } p_i p_j \text{ Chain Edge.} \end{cases}$$



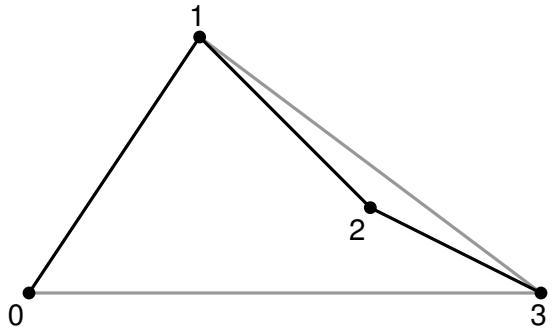
	1	2	3
0	0	-	-
1		0	+
2			0

- ermöglicht Vergleich von Chains aufgrund ihrer $V(C)$
- $V(C_1) = V(C_2) \Rightarrow \text{tr}(C_1) = \text{tr}(C_2)$

Strukturelle Eigenschaften von Chains - Visibility Triangle

- zu jeder Chain gibt es ein *Visibility Triangle* $V(C)$:

$$V(C)_{i,j} = \begin{cases} +1, & \text{falls } p_i p_j \text{ über Chain Curve (}\textit{obere Kante}\text{)}, \\ -1, & \text{falls } p_i p_j \text{ unter Chain Curve (}\textit{untere Kante}\text{)}, \\ 0, & \text{falls } p_i p_j \text{ Chain Edge.} \end{cases}$$



	1	2	3
0	0	-	-
1		0	+
2			0

- ermöglicht Vergleich von Chains aufgrund ihrer $V(C)$
- $V(C_1) = V(C_2) \Rightarrow \text{tr}(C_1) = \text{tr}(C_2)$
- Chain nach oben / unten geöffnet $\rightarrow$ *Upward* / *Downward* Chain
  - vgl. Eintrag in Zelle $(0, n)$ von $V(C)$

Strukturelle Eigenschaften - Flips

- zu einer Chain C gibt es eine *geflippte* Variante $\overline{C}$, für deren Visibility Triangle gilt:

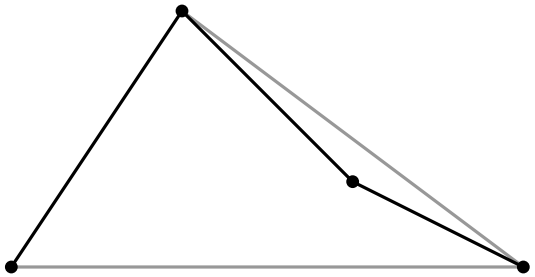
$$V(\overline{C}) = -V(C)$$

Strukturelle Eigenschaften - Flips

- zu einer Chain C gibt es eine *geflippte* Variante $\overline{C}$, für deren Visibility Triangle gilt:

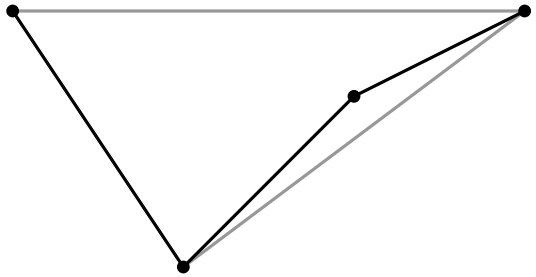
$$V(\overline{C}) = -V(C)$$

C



0 - -
0 +
0

$\overline{C}$



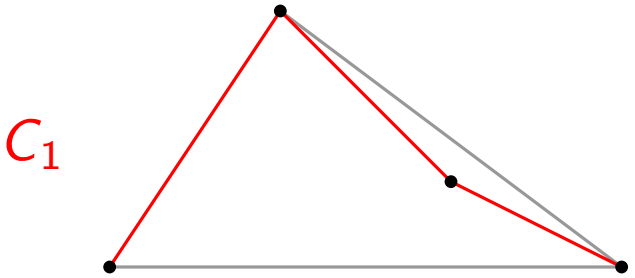
0 + +
0 -
0

Strukturelle Eigenschaften - Konvexe Summe

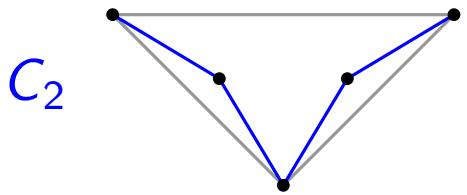
- Um zwei Chains C_1, C_2 der Länge n_1 bzw. n_2 zu konkatenieren, können folgende Operationen verwendet werden:

Strukturelle Eigenschaften - Konvexe Summe

- Um zwei Chains C_1, C_2 der Länge n_1 bzw. n_2 zu konkatenieren, können folgende Operationen verwendet werden:
- *Konvexe Summe* $C_1 \vee C_2$:



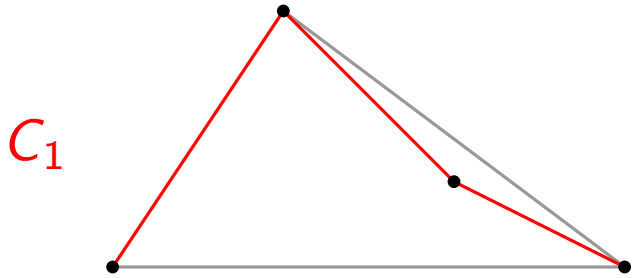
0	-	-
	0	+
		0



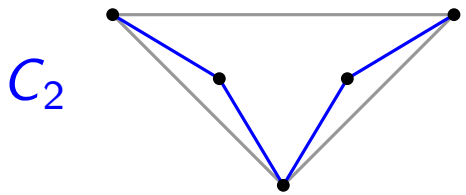
0	-	+	+
	0	+	+
		0	-
			0

Strukturelle Eigenschaften - Konvexe Summe

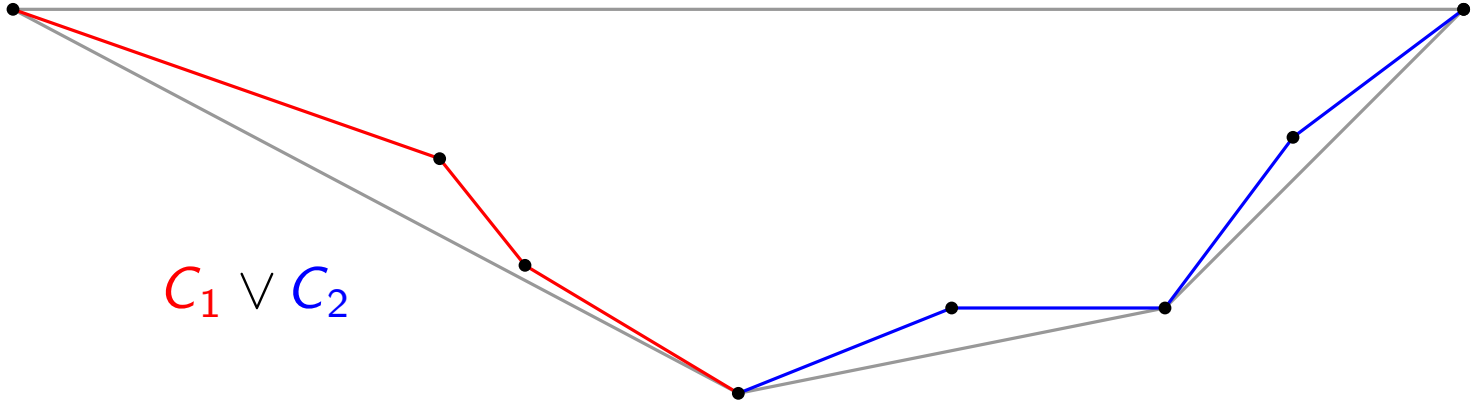
- Um zwei Chains C_1, C_2 der Länge n_1 bzw. n_2 zu konkatenieren, können folgende Operationen verwendet werden:
- *Konvexe Summe* $C_1 \vee C_2$:



	-	-
0		
	0	+
		0



0	-	+	+
	0	+	+
		0	-
			0



0	-	-	+	+	+	+
	0	+	+	+	+	+
		0	+	+	+	+
			0	-	+	+
				0	+	+
					0	-
						0

Strukturelle Eigenschaften - Konvexe Summe

- Um zwei Chains C_1, C_2 der Länge n_1 bzw. n_2 zu konkatenieren, können folgende Operationen verwendet werden:
- *Konvexe Summe* $C_1 \vee C_2$:

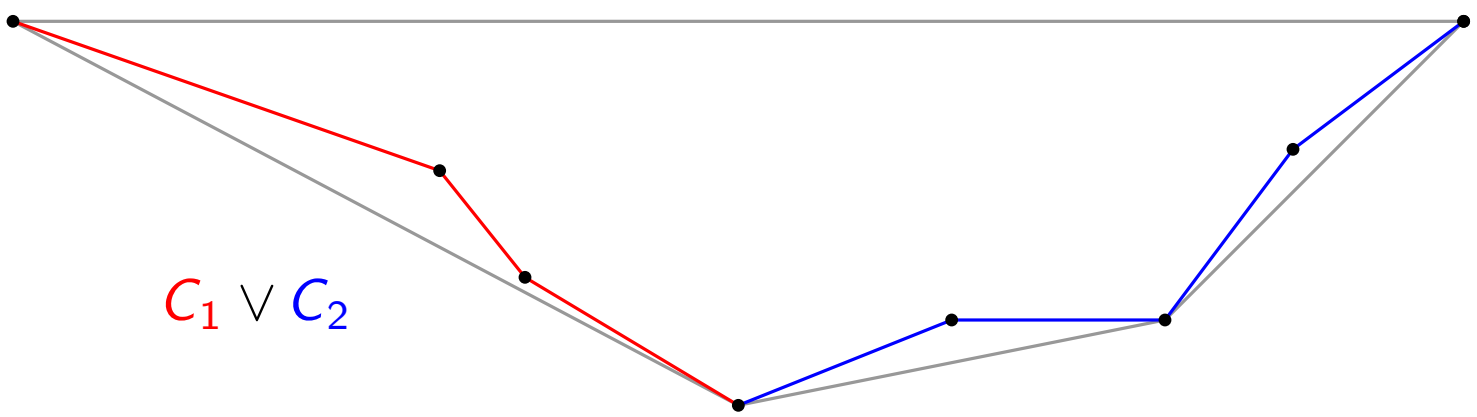
$$V(C_1 \vee C_2)_{i,j} = \begin{cases} V(C_1)_{i,j}, & \text{falls } i,j \in [0, n_1], \\ V(C_2)_{i-n_1,j-n_1}, & \text{falls } i,j \in [n_1, n_1+n_2], \\ +1, & \text{falls } i < n_1 < j. \end{cases}$$

C_1

	0	-	-
0		0	+
			0

C_2

	0	-	+	+
0		0	+	+
			0	-
				0

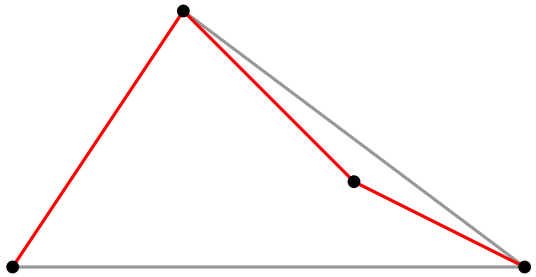


	0	-	-	+	+	+	+
0		0	+	+	+	+	+
			0	+	+	+	+
				0	-	+	+
					0	+	+
						0	-
							0

Strukturelle Eigenschaften - Konkave Summe

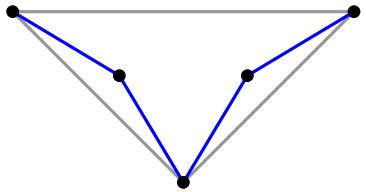
- *Konkave Summe* $C_1 \wedge C_2$:

C_1



0 - -
0 +
0

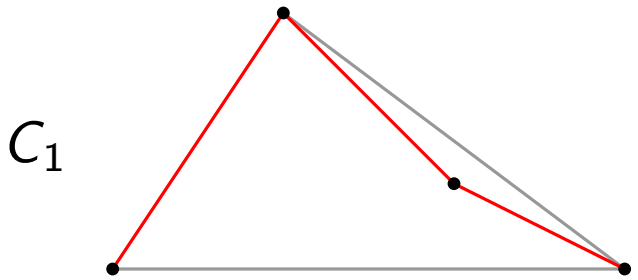
C_2



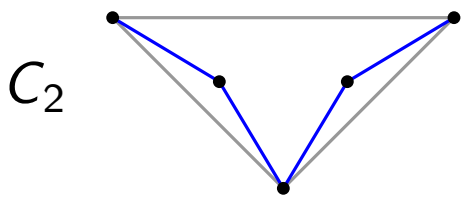
0 - + +
0 + +
0 -
0

Strukturelle Eigenschaften - Konkave Summe

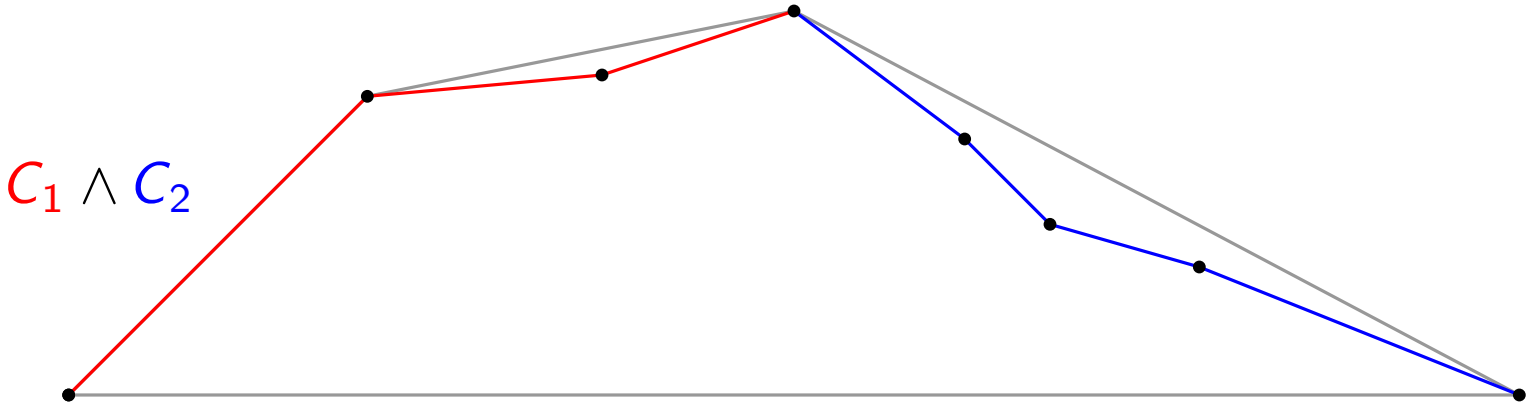
- *Konkave Summe* $C_1 \wedge C_2$:



0 - -
0 +
0

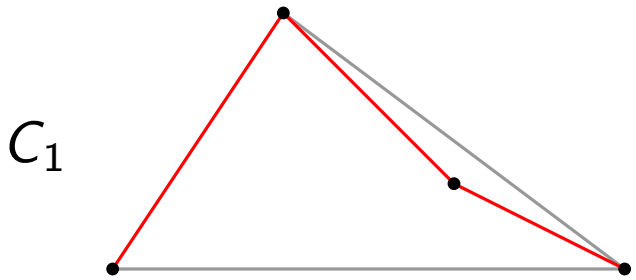


0 - + +
0 + +
0 -
0

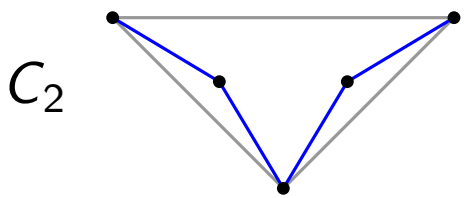


Strukturelle Eigenschaften - Konkave Summe

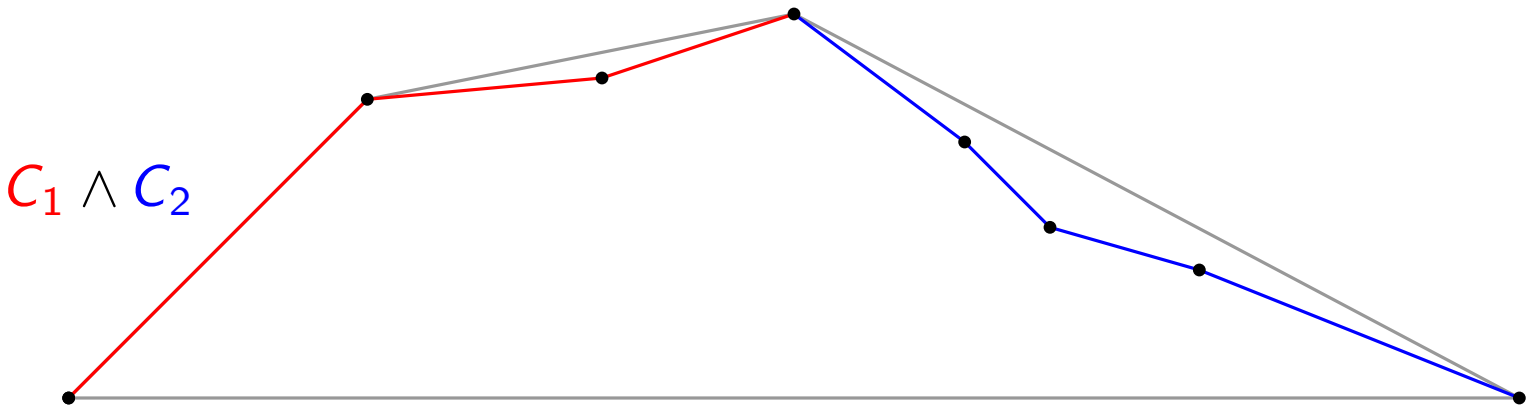
- *Konkave Summe* $C_1 \wedge C_2$:



0 - -
0 +
0



0 - + +
0 + +
0 -
0

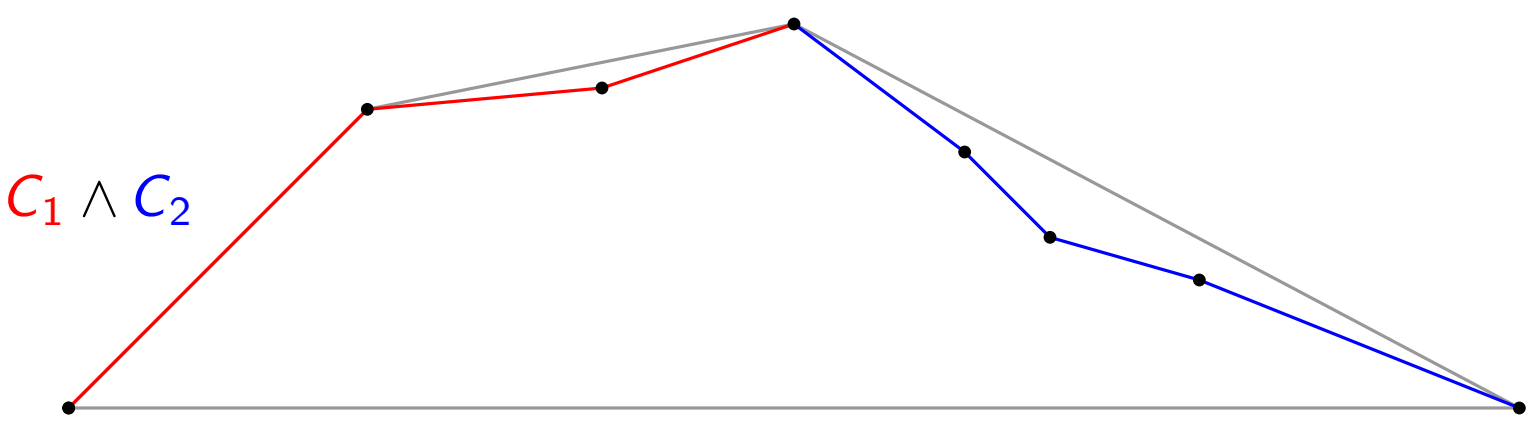


0 - - - - - -
0 + - - - -
0 - - - -
0 - + +
0 + +
0 -
0

Strukturelle Eigenschaften - Konkave Summe

- *Konkave Summe* $C_1 \wedge C_2$:

$$V(C_1 \wedge C_2)_{i,j} = \begin{cases} V(C_1)_{i,j}, & \text{falls } i,j \in [0, n_1], \\ V(C_2)_{i-n_1,j-n_1}, & \text{falls } i,j \in [n_1, n_1+n_2], \\ -1, & \text{falls } i < n_1 < j. \end{cases}$$

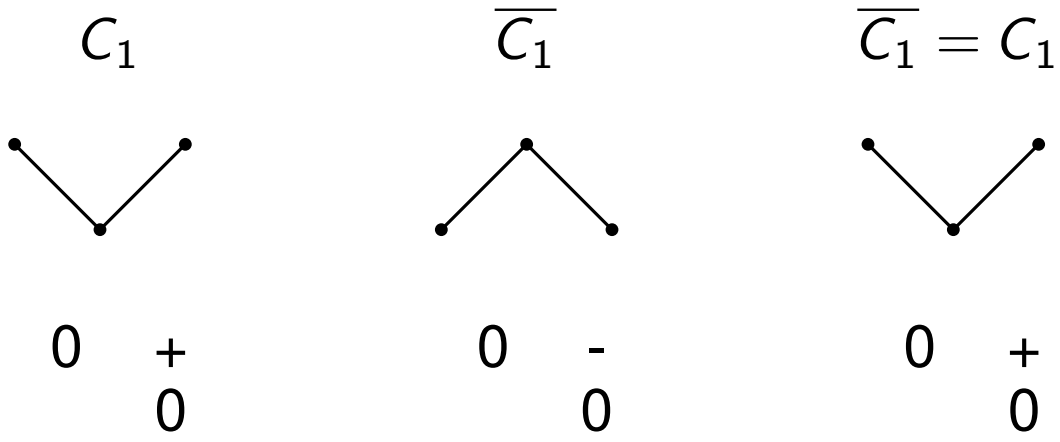


C_1			C_2			
			0	-	+	+
0	-	-		0	+	+
	0	+			0	-
		0				0

0	-	-	-	-	-	-
	0	+	-	-	-	-
		0	-	-	-	-
			0	-	+	+
				0	+	+
					0	-
						0

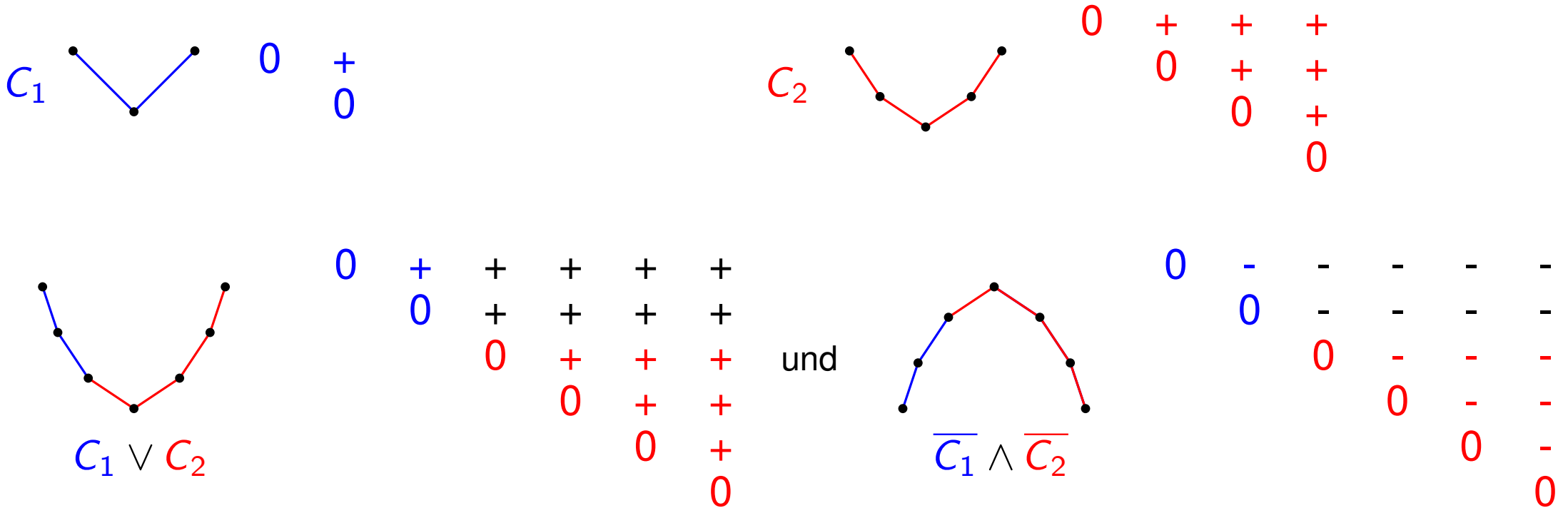
Strukturelle Eigenschaften - Algebraische Eigenschaften

- *Involution*: $V(\overline{\overline{C_1}}) = V(C_1)$



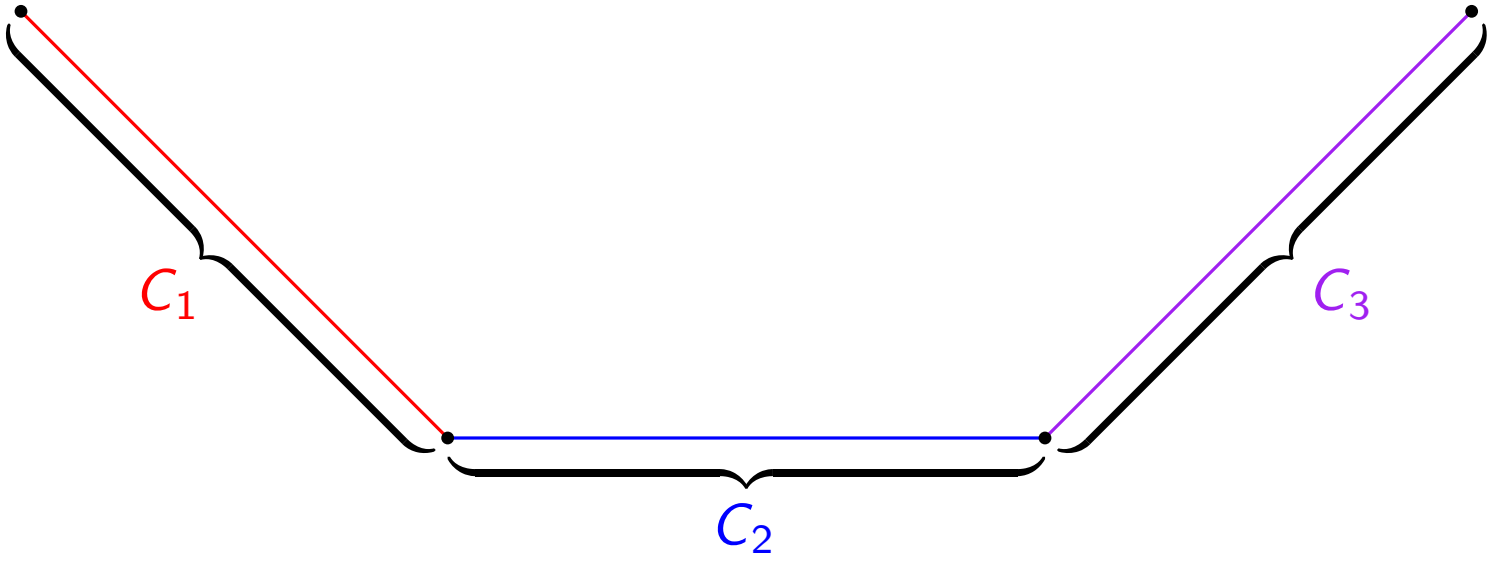
Strukturelle Eigenschaften - Algebraische Eigenschaften

- *Involution*: $V(\overline{\overline{C_1}}) = V(C_1)$
- *De Morgan*: $V(\overline{C_1 \vee C_2}) = V(\overline{C_1} \wedge \overline{C_2})$ und $V(\overline{C_1 \wedge C_2}) = V(\overline{C_1} \vee \overline{C_2})$



Strukturelle Eigenschaften - Algebraische Eigenschaften

- *Involution*: $V(\overline{\overline{C_1}}) = V(C_1)$
- *De Morgan*: $V(\overline{C_1 \vee C_2}) = V(\overline{C_1} \wedge \overline{C_2})$ und $V(\overline{C_1 \wedge C_2}) = V(\overline{C_1} \vee \overline{C_2})$
- *Assoziativität*: $V((C_1 \vee C_2) \vee C_3) = V(C_1 \vee (C_2 \vee C_3))$ und $V((C_1 \wedge C_2) \wedge C_3) = V(C_1 \wedge (C_2 \wedge C_3))$



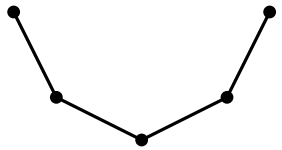
Strukturelle Eigenschaften - Beispiele

- durch Techniken kann man nun verschiedene Chains bauen

Strukturelle Eigenschaften - Beispiele

- durch Techniken kann man nun verschiedene Chains bauen
- $C_{cvx}(n) = \underbrace{E \vee \cdots \vee E}_{n \text{ Kopien}}$

$C_{cvx}(4)$:

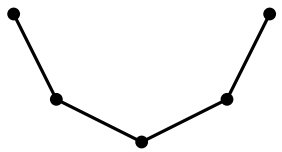


$$\begin{array}{ccccc} 0 & + & + & + \\ & 0 & + & + \\ & & 0 & + \\ & & & 0 \end{array}$$

Strukturelle Eigenschaften - Beispiele

- durch Techniken kann man nun verschiedene Chains bauen
- $C_{cvx}(n) = \underbrace{E \vee \cdots \vee E}_{n \text{ Kopien}}$

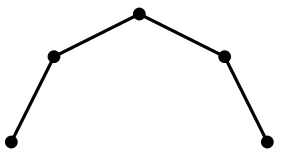
$C_{cvx}(4)$:



0 + + +
0 + +
0 +
0

- $C_{ccv}(n) = \underbrace{E \wedge \cdots \wedge E}_{n \text{ Kopien}}$

$C_{ccv}(4)$:

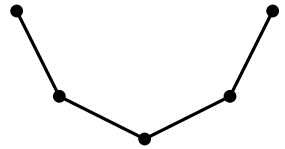


0 - - -
0 - -
0 -
0

Strukturelle Eigenschaften - Beispiele

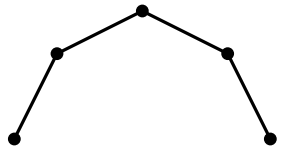
- durch Techniken kann man nun verschiedene Chains bauen
- $C_{cvx}(n) = \underbrace{E \vee \cdots \vee E}_{n \text{ Kopien}}$
- $C_{ccv}(n) = \underbrace{E \wedge \cdots \wedge E}_{n \text{ Kopien}}$

$C_{cvx}(4)$:



$$\begin{array}{cccc} 0 & + & + & + \\ & 0 & + & + \\ & & 0 & + \\ & & & 0 \end{array}$$

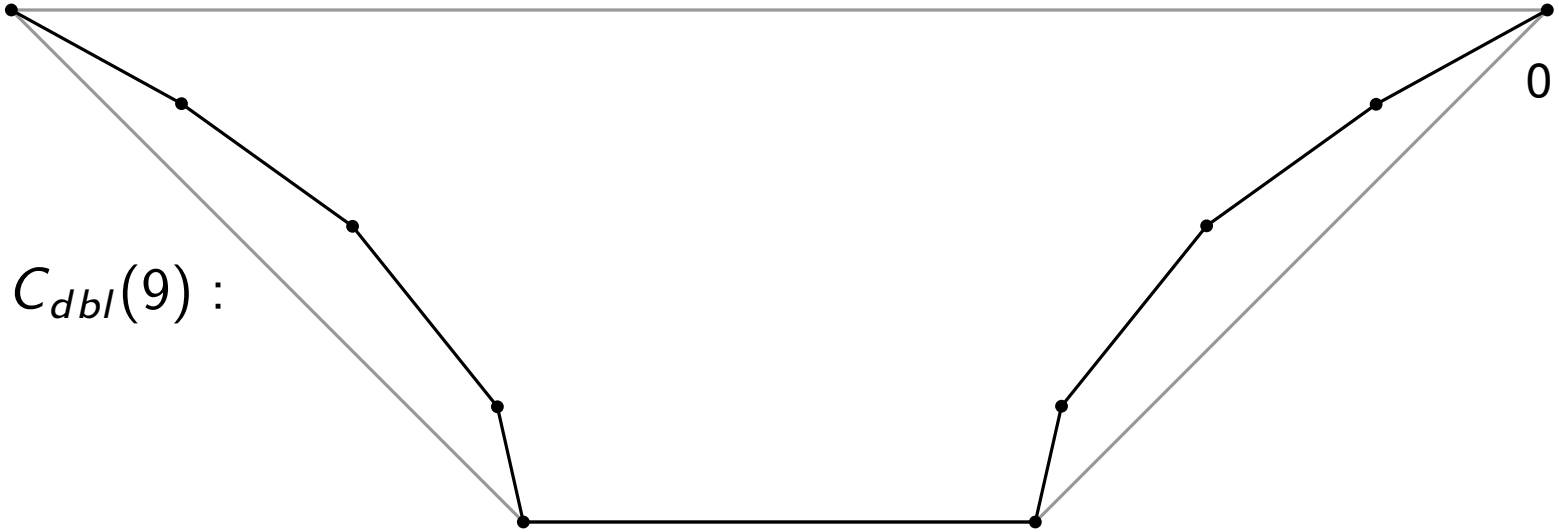
$C_{ccv}(4)$:



$$\begin{array}{cccc} 0 & - & - & - \\ & 0 & - & - \\ & & 0 & - \\ & & & 0 \end{array}$$

- *Double Chain* $C_{dbl}(n) = C_{ccv}(k) \vee E \vee C_{ccv}(k)$

$C_{dbl}(9)$:



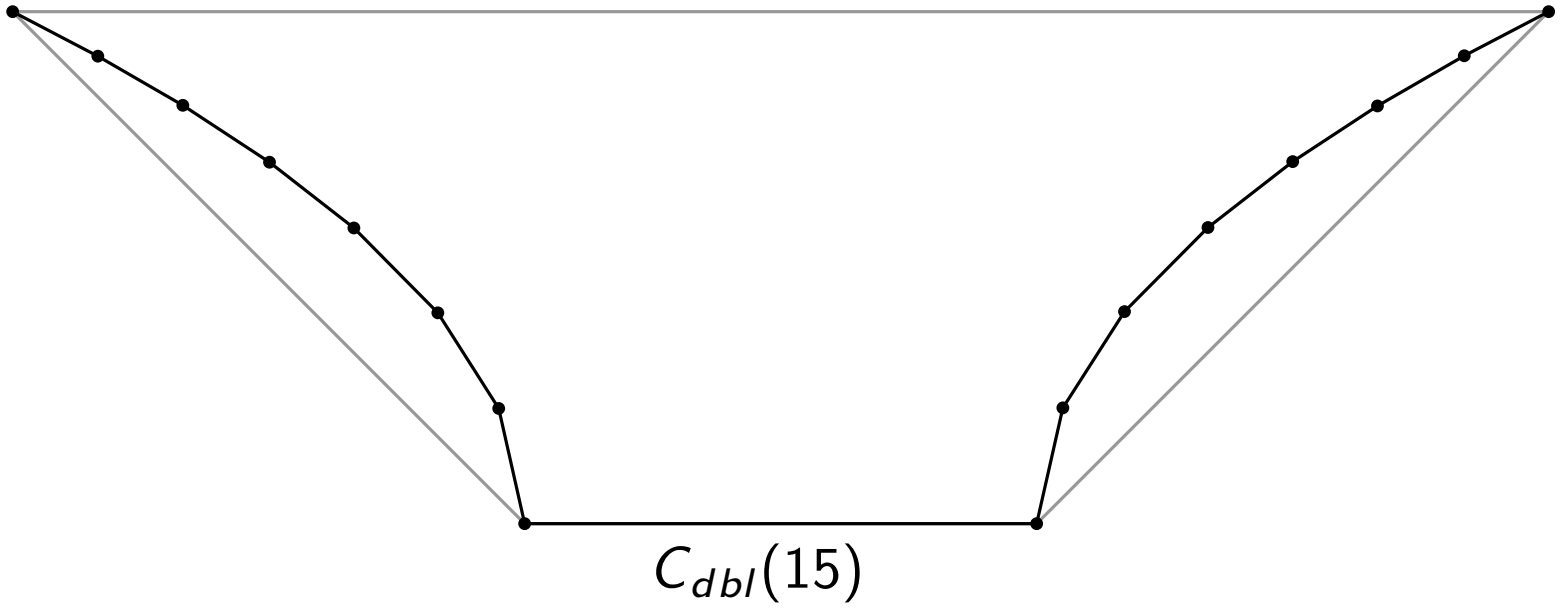
$$\begin{array}{ccccccccc} 0 & - & - & - & + & + & + & + & + \\ & 0 & - & - & + & + & + & + & + \\ & & 0 & - & + & + & + & + & + \\ & & & 0 & + & + & + & + & + \\ & & & & 0 & + & + & + & + \\ & & & & & 0 & - & - & - \\ & & & & & & 0 & - & - \\ & & & & & & & 0 & - \\ & & & & & & & & 0 \end{array}$$

Triangulierung von Chains - Oben und unten

- Chain Edges teilen Triangulierung in *obere* und *untere Triangulierung*
- können somit separat analysiert werden
- Anzahl oberer und unterer Triangulierungen $U(C)$ und $L(C)$, für die gilt: $tr(C) = U(C) \cdot L(C)$

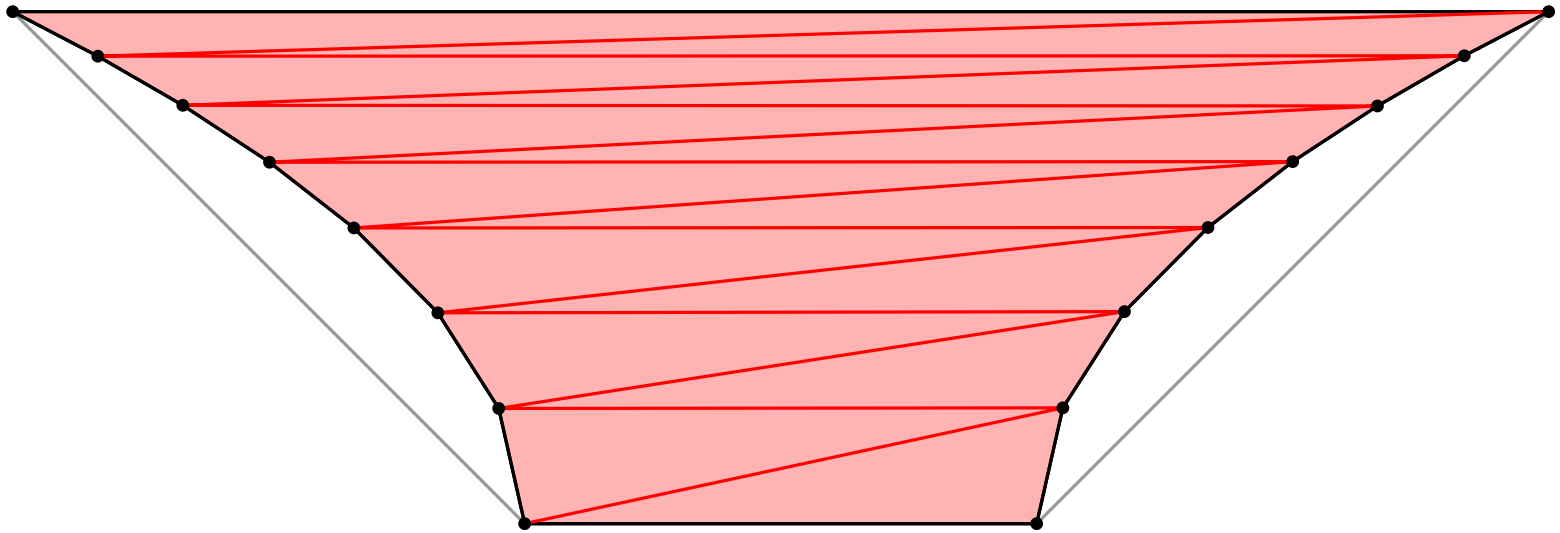
Triangulierung von Chains - Oben und unten

- Chain Edges teilen Triangulierung in *obere* und *untere Triangulierung*
- können somit separat analysiert werden
- Anzahl oberer und unterer Triangulierungen $U(C)$ und $L(C)$, für die gilt: $tr(C) = U(C) \cdot L(C)$



Triangulierung von Chains - Oben und unten

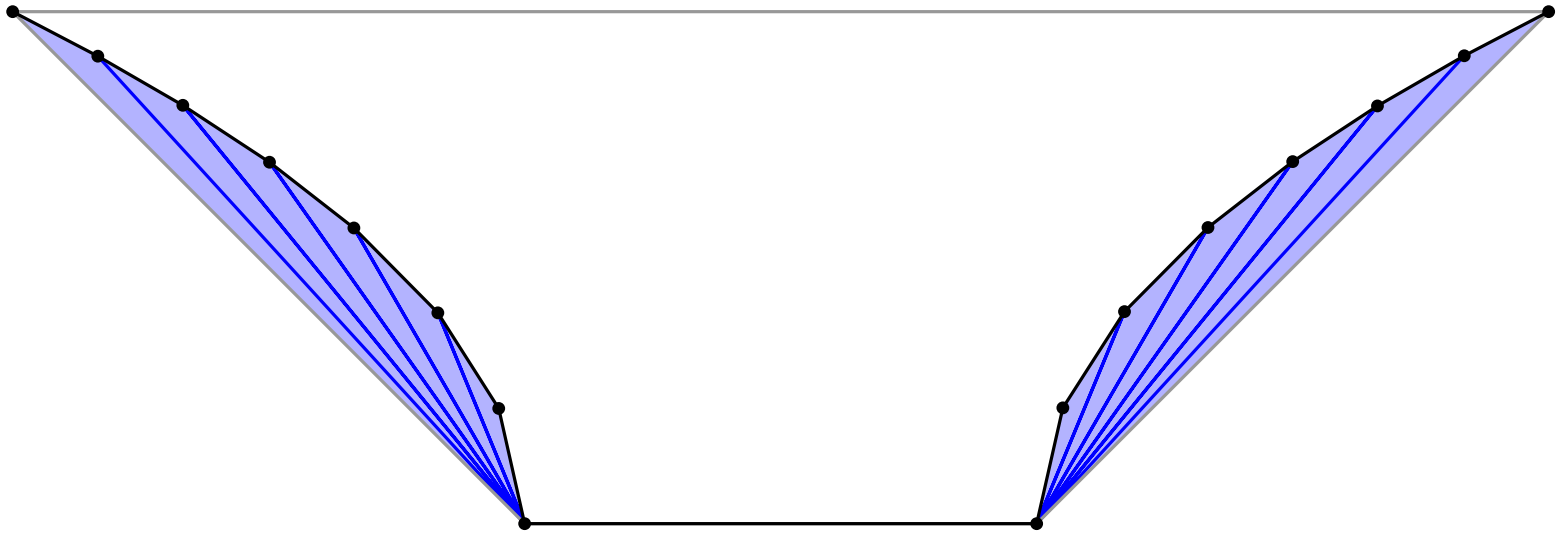
- Chain Edges teilen Triangulierung in *obere* und *untere Triangulierung*
- können somit separat analysiert werden
- Anzahl oberer und unterer Triangulierungen $U(C)$ und $L(C)$, für die gilt: $tr(C) = U(C) \cdot L(C)$



obere Triangulierung

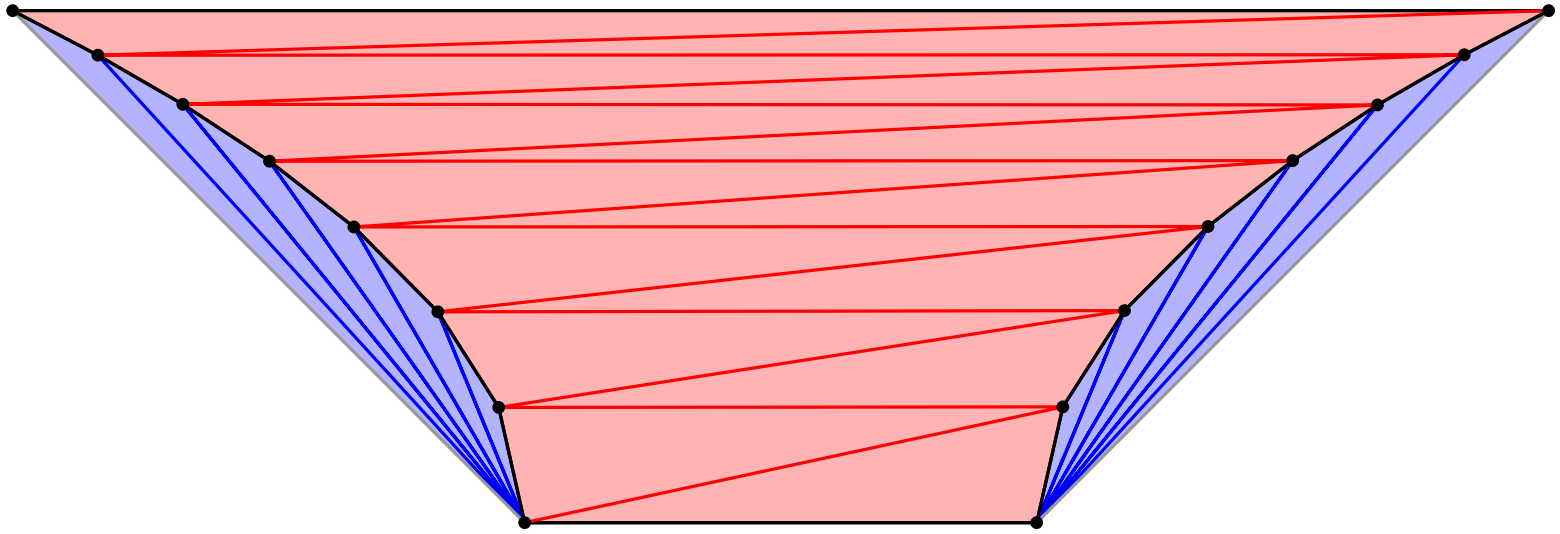
Triangulierung von Chains - Oben und unten

- Chain Edges teilen Triangulierung in *obere* und *untere Triangulierung*
- können somit separat analysiert werden
- Anzahl oberer und unterer Triangulierungen $U(C)$ und $L(C)$, für die gilt: $tr(C) = U(C) \cdot L(C)$



Triangulierung von Chains - Oben und unten

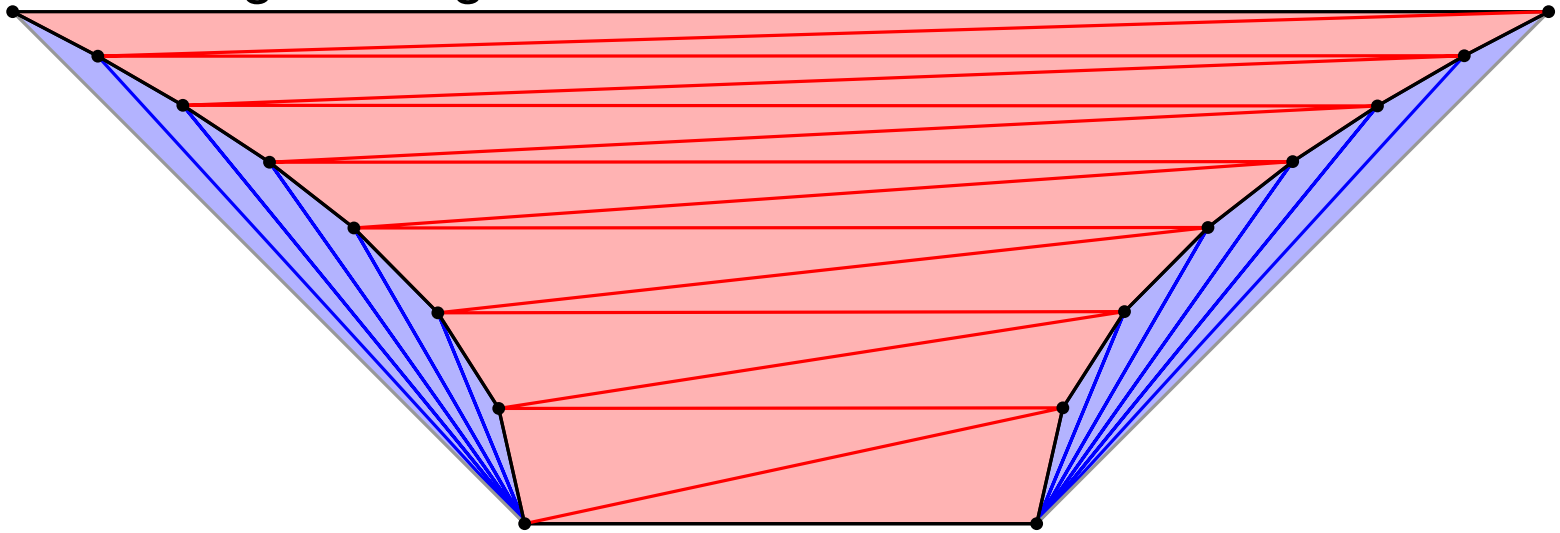
- Chain Edges teilen Triangulierung in *obere* und *untere Triangulierung*
- können somit separat analysiert werden
- Anzahl oberer und unterer Triangulierungen $U(C)$ und $L(C)$, für die gilt: $tr(C) = U(C) \cdot L(C)$



komplette Triangulierung

Triangulierung von Chains - Oben und unten

- Chain Edges teilen Triangulierung in *obere* und *untere Triangulierung*
- können somit separat analysiert werden
- Anzahl oberer und unterer Triangulierungen $U(C)$ und $L(C)$, für die gilt: $tr(C) = U(C) \cdot L(C)$
- beachte $L(C) = U(\overline{C})$, daher reicht im Folgenden Betrachtung der Eigenschaften von oberen Triangulierungen aus



komplette Triangulierung

Triangulierung von Chains - Partielle obere Triangulierung

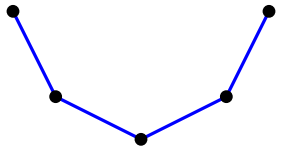
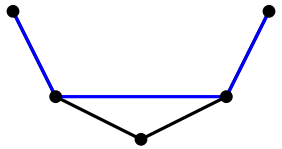
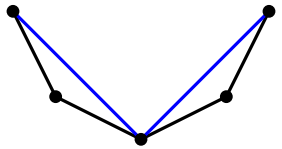
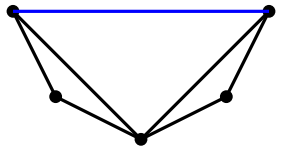
- wollen Vorgang der Triangulierung formalisieren
- dazu bauen wir ab jetzt obere Triangulierungen von den Chain Edges an nach oben hin auf

Triangulierung von Chains - Partielle obere Triangulierung

- wollen Vorgang der Triangulierung formalisieren
- dazu bauen wir ab jetzt obere Triangulierungen von den Chain Edges an nach oben hin auf
- zu jeder Chain C lassen sich so *partielle obere Triangulierungen* (PUT) mit von oben *sichtbaren Kanten* V bilden
- hat bei k eingefügten Dreiecken $n - k$ sichtbare Kanten

Triangulierung von Chains - Partielle obere Triangulierung

- wollen Vorgang der Triangulierung formalisieren
- dazu bauen wir ab jetzt obere Triangulierungen von den Chain Edges an nach oben hin auf
- zu jeder Chain C lassen sich so *partielle obere Triangulierungen* (PUT) mit von oben *sichtbaren Kanten V* bilden
- hat bei k eingefügten Dreiecken $n - k$ sichtbare Kanten

$\lvert V \rvert$	4	3	2	1
k	0	1	2	3
$C_{cvx}(4)$				

Triangulierung von Chains - Triangulierungspolynom

- sei $t_k(C)$ die *Anzahl partieller oberer Triangulierungen* mit k Dreiecken
- dann ist das obere *Triangulierungspolynom* (TP) von C die korrespondierende erzeugende Funktion:

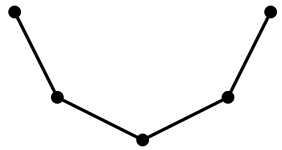
$$T_C(x) = \sum_{k=0}^{n-1} t_k(C) x^k$$

Triangulierung von Chains - Triangulierungspolynom

- sei $t_k(C)$ die *Anzahl partieller oberer Triangulierungen* mit k Dreiecken
- dann ist das obere *Triangulierungspolynom* (TP) von C die korrespondierende erzeugende Funktion:

$$T_C(x) = \sum_{k=0}^{n-1} t_k(C) x^k$$

- Beispiel $C_{cvx}(4)$: $T_{C_{cvx}(4)}(x) =$



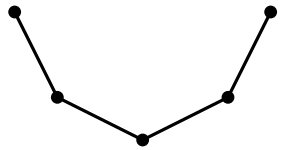
Triangulierung von Chains - Triangulierungspolynom

- sei $t_k(C)$ die *Anzahl partieller oberer Triangulierungen* mit k Dreiecken
- dann ist das obere *Triangulierungspolynom* (TP) von C die korrespondierende erzeugende Funktion:

$$T_C(x) = \sum_{k=0}^{n-1} t_k(C)x^k$$

- Beispiel $C_{cvx}(4)$: $T_{C_{cvx}(4)}(x) = 1+$

$k = 0$



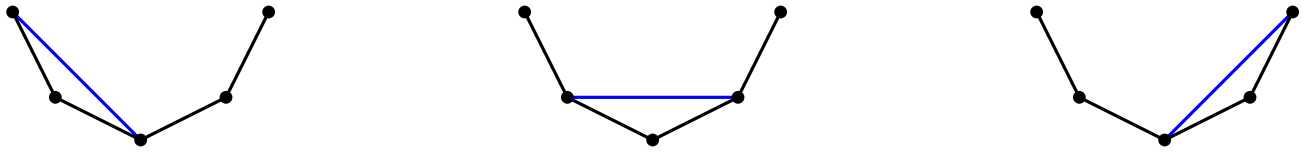
Triangulierung von Chains - Triangulierungspolynom

- sei $t_k(C)$ die *Anzahl partieller oberer Triangulierungen* mit k Dreiecken
- dann ist das obere *Triangulierungspolynom* (TP) von C die korrespondierende erzeugende Funktion:

$$T_C(x) = \sum_{k=0}^{n-1} t_k(C)x^k$$

- Beispiel $C_{cvx}(4)$: $T_{C_{cvx}(4)}(x) = 1 + 3x +$

$k = 1$



Triangulierung von Chains - Triangulierungspolynom

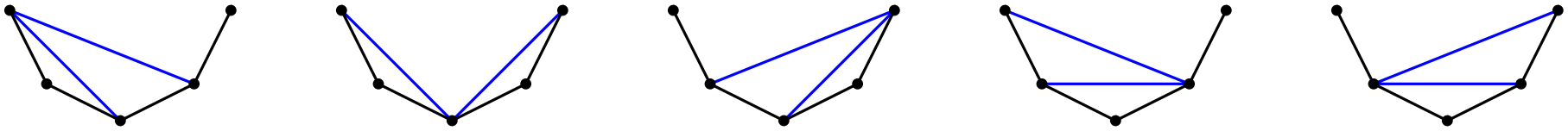
- sei $t_k(C)$ die *Anzahl partieller oberer Triangulierungen* mit k Dreiecken
- dann ist das obere *Triangulierungspolynom* (TP) von C die korrespondierende erzeugende Funktion:

$$T_C(x) = \sum_{k=0}^{n-1} t_k(C) x^k$$

- Beispiel $C_{cvx}(4)$:

$$T_{C_{cvx}(4)}(x) = 1 + 3x + 5x^2$$

$k = 2$



Triangulierung von Chains - Triangulierungspolynom

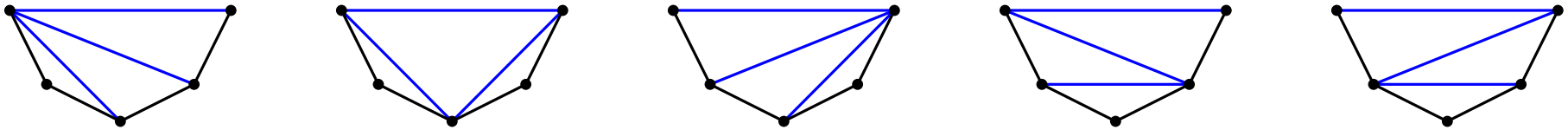
- sei $t_k(C)$ die *Anzahl partieller oberer Triangulierungen* mit k Dreiecken
- dann ist das obere *Triangulierungspolynom* (TP) von C die korrespondierende erzeugende Funktion:

$$T_C(x) = \sum_{k=0}^{n-1} t_k(C) x^k$$

- Beispiel $C_{cvx}(4)$:

$$T_{C_{cvx}(4)}(x) = 1 + 3x + 5x^2 + 5x^3$$

$k = 3$



Triangulierung von Chains - TP und konkave Summe

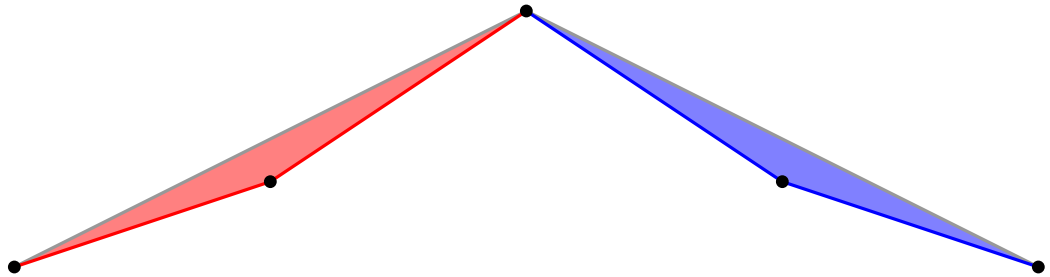
- Was passiert, mit dem TP wenn man zwei Chains nun mittels konvexer oder konkaver Summen konkateniert?

Triangulierung von Chains - TP und konkave Summe

- Was passiert, mit dem TP wenn man zwei Chains nun mittels konvexer oder konkaver Summen konkateniert?
- konkaver Fall einfach:

$$T_{C_1 \wedge C_2}(x) = T_{C_1}(x) \cdot T_{C_2}(x) \qquad \text{und} \qquad U(C_1 \wedge C_2) = U(C_1) \cdot U(C_2)$$

- bei konkaver Summe gibt es keine Möglichkeit Triangulierungen der beiden Summanden zu verbinden



Triangulierung von Chains - TP und konvexe Summe

- konvexer Fall hingegen komplexer für obere Triangulierungen
- jede PUT von $C_1 \vee C_2$ besteht aus einer PUT von C_1, einer PUT von C_2 und Kanten zwischen C_1 und C_2

Triangulierung von Chains - TP und konvexe Summe

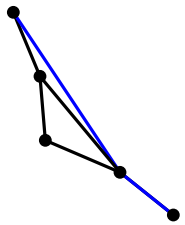
- konvexer Fall hingegen komplexer für obere Triangulierungen
- jede PUT von $C_1 \vee C_2$ besteht aus einer PUT von C_1, einer PUT von C_2 und Kanten zwischen C_1 und C_2
- genauer ergibt sich:

$$T_{C_1 \vee C_2}(x) = \sum_{k_1=0}^{n_1-1} \sum_{k_2=0}^{n_2-1} t_{k_1}(C_1) \cdot t_{k_2}(C_2) \cdot x^{k_1+k_2} \cdot T_{C_{ccv}(n_1-k_1) \vee C_{ccv}(n_2-k_2)}(x)$$

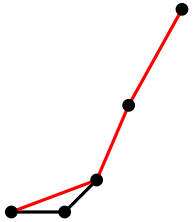
Triangulierung von Chains - TP und konvexe Summe

- konvexer Fall hingegen komplexer für obere Triangulierungen
- jede PUT von $C_1 \vee C_2$ besteht aus einer PUT von C_1, einer PUT von C_2 und Kanten zwischen C_1 und C_2
- genauer ergibt sich:

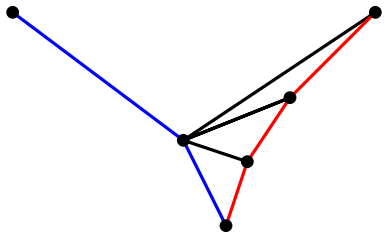
$$T_{C_1 \vee C_2}(x) = \sum_{k_1=0}^{n_1-1} \sum_{k_2=0}^{n_2-1} t_{k_1}(C_1) \cdot t_{k_2}(C_2) \cdot x^{k_1+k_2} \cdot T_{C_{ccv}(n_1-k_1) \vee C_{ccv}(n_2-k_2)}(x)$$



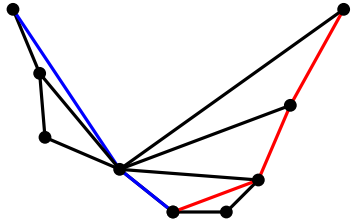
C_1



C_2



$C_{ccv}(2) \vee C_{ccv}(3)$

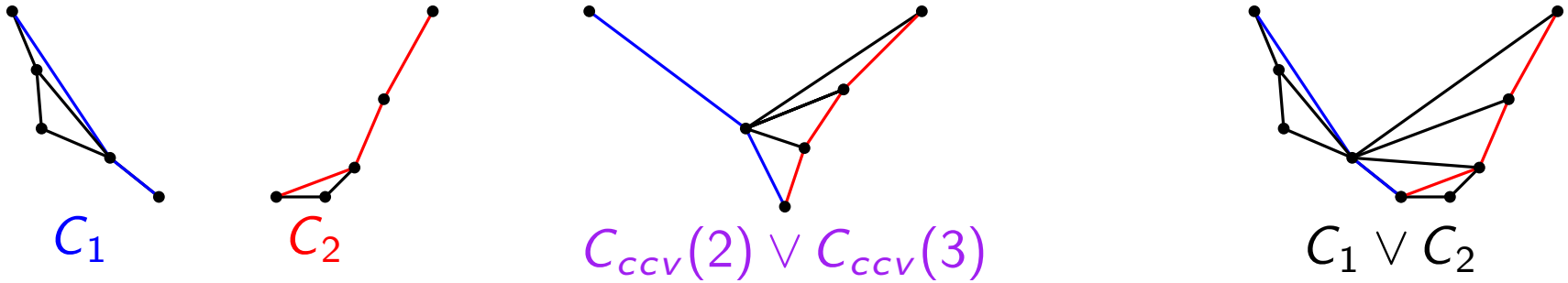


$C_1 \vee C_2$

Triangulierung von Chains - TP und konvexe Summe

- konvexer Fall hingegen komplexer für obere Triangulierungen
- jede PUT von $C_1 \vee C_2$ besteht aus einer PUT von C_1, einer PUT von C_2 und Kanten zwischen C_1 und C_2
- genauer ergibt sich:

$$T_{C_1 \vee C_2}(x) = \sum_{k_1=0}^{n_1-1} \sum_{k_2=0}^{n_2-1} t_{k_1}(C_1) \cdot t_{k_2}(C_2) \cdot x^{k_1+k_2} \cdot T_{C_{ccv}(n_1-k_1) \vee C_{ccv}(n_2-k_2)}(x)$$



- da Chains nur aus konvexen bzw. konkaven Summen der primitiven Chain bestehen, kann somit auch das TP für jede Chain berechnet werden

Koch Chain

Vorstellung des neuen Rekordhalters

Koch Chain

Vorstellung des neuen Rekordhalters

- inspiriert von Koch Kurve: Fraktal, stetige, aber an keiner Stelle differenzierbare Kurve

Koch Chain

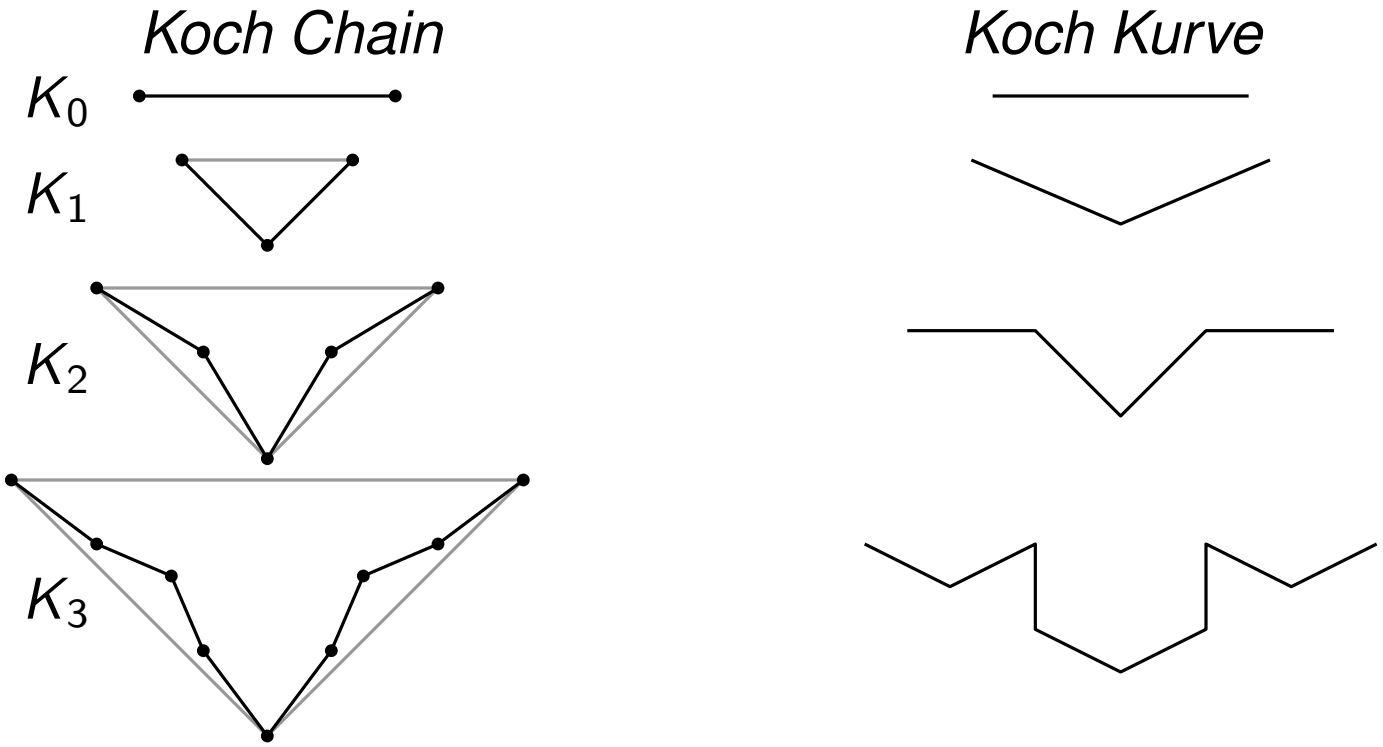
Vorstellung des neuen Rekordhalters

- inspiriert von Koch Kurve: Fraktal, stetige, aber an keiner Stelle differenzierbare Kurve
- rekursiv definiert: $K_s = \overline{K_{s-1}} \vee \overline{K_{s-1}}$ mit $n = 2^s$ Chain Edges

Koch Chain

Vorstellung des neuen Rekordhalters

- inspiriert von Koch Kurve: Fraktal, stetige, aber an keiner Stelle differenzierbare Kurve
- rekursiv definiert: $K_s = \overline{K_{s-1}} \vee \overline{K_{s-1}}$ mit $n = 2^s$ Chain Edges



Koch Chain Auswertung

- K_3 hat 424 verschiedene Triangulierungen (4 untere und 106 obere)
- beim K_4 sind es schon ca. 975 Millionen

Koch Chain Auswertung

- K_3 hat 424 verschiedene Triangulierungen (4 untere und 106 obere)
- beim K_4 sind es schon ca. 975 Millionen
- dadurch experimentelle Analyse schnell rechenaufwändig
- konvergiert aber relativ schnell in Richtung $\Omega(9.08^n)$

Koch Chain Auswertung

- K_3 hat 424 verschiedene Triangulierungen (4 untere und 106 obere)
- beim K_4 sind es schon ca. 975 Millionen
- dadurch experimentelle Analyse schnell rechenaufwändig
- konvergiert aber relativ schnell in Richtung $\Omega(9.08^n)$

Durch das Ausnutzen der strukturellen Eigenschaften von Chains und der dadurch möglichen Benutzung von erzeugenden Funktionen, lassen sich folgende Schranken beweisen:

$$9.082799^n \leq \lim_{s \to \infty} \mathrm{tr}(K_s) \leq 9.083139^n$$

Was haben wir gesehen?

- durch Chain-Konzept lassen sich viele relevante Punktmengen darstellen

Was haben wir gesehen?

- durch Chain-Konzept lassen sich viele relevante Punktmengen darstellen
- strukturelle Eigenschaften von Chains ermöglichen Darstellung jeder Chain nur durch konvexe und konkave Summen

Was haben wir gesehen?

- durch Chain-Konzept lassen sich viele relevante Punktmengen darstellen
- strukturelle Eigenschaften von Chains ermöglichen Darstellung jeder Chain nur durch konvexe und konkave Summen
- dadurch möglich, für jede Chain eine erzeugende Funktion zu berechnen

Was haben wir gesehen?

- durch Chain-Konzept lassen sich viele relevante Punktmengen darstellen
- strukturelle Eigenschaften von Chains ermöglichen Darstellung jeder Chain nur durch konvexe und konkave Summen
- dadurch möglich, für jede Chain eine erzeugende Funktion zu berechnen
- mit Hilfe dieser erzeugenden Funktionen lassen sich präzise Schranken für die Anzahl verschiedener Triangulierungen der Chains berechnen

Was haben wir gesehen?

- durch Chain-Konzept lassen sich viele relevante Punktmengen darstellen
- strukturelle Eigenschaften von Chains ermöglichen Darstellung jeder Chain nur durch konvexe und konkave Summen
- dadurch möglich, für jede Chain eine erzeugende Funktion zu berechnen
- mit Hilfe dieser erzeugenden Funktionen lassen sich präzise Schranken für die Anzahl verschiedener Triangulierungen der Chains berechnen
- Punktmengenfamilie der Koch Chains erreicht deutlich höheren Wert an verschiedenen Triangulierungen als die zuvor führenden Generalized Double Zig-Zag Chains