

# Vorlesung Algorithmische Kartografie

## Punktbeschriftung in Dynamischen Karten

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Benjamin Niedermann · **Martin Nöllenburg**  
28.05.2015



# Übungen – Nachtrag

- 1) Überlegen Sie sich, wie man den  $O(n \log n)$ -Algorithmus auf Label mit unterschiedlichen Höhen anpassen kann. Ändert sich dadurch die Laufzeit?

**while**  $\mathcal{H}_{\text{right}}$ ,  $\mathcal{H}_{\text{int}}$  oder  $\mathcal{H}_V$  nicht leer **do**

$v \leftarrow \text{find-min}(\mathcal{H}_V)$

**while**  $v$  links von  $F$  **do**

        | delete-min( $\mathcal{H}_V$ );  $v \leftarrow \text{find-min}(\mathcal{H}_V)$

$l_i \leftarrow$  linkstes Label aus  $\mathcal{H}_{\text{right}}$ ,  $\mathcal{H}_{\text{int}}$  und  $\mathcal{H}_V$

    füge  $l_i$  zur Beschriftung hinzu

$f_{\text{new}} \leftarrow$  rechte Kante von  $\tilde{l}_i$

    update  $F$  und  $\mathcal{T}_V$  mit  $f_{\text{new}}$

    suche mit der Region links von  $f_{\text{new}}$  in  $\mathcal{P}_{\text{left}}$  und  $\mathcal{P}_{\text{right}}$  und update

$\mathcal{H}_{\text{right}}$ ,  $\mathcal{P}_{\text{left}}$ ,  $\mathcal{H}_{\text{int}}$ ,  $\mathcal{T}_i$ 's und  $\mathcal{P}_{\text{right}}$  wie beschrieben

    entferne alle Verweise auf  $p_i$  aus den Datenstrukturen

# Übungen – Nachtrag

- 1) Überlegen Sie sich, wie man den  $O(n \log n)$ -Algorithmus auf Label mit unterschiedlichen Höhen anpassen kann. Ändert sich dadurch die Laufzeit?

**while**  $\mathcal{H}_{\text{right}}$ ,  $\mathcal{H}_{\text{int}}$  oder  $\mathcal{H}_V$  nicht leer **do**

$v \leftarrow \text{find-min}(\mathcal{H}_V)$

**while**  $v$  links von  $F$  **do**

$\lfloor$  delete-min( $\mathcal{H}_V$ );  $v \leftarrow \text{find-min}(\mathcal{H}_V)$

$l_i \leftarrow$  linkestes Label aus  $\mathcal{H}_{\text{right}}$ ,  $\mathcal{H}_{\text{int}}$  und  $\mathcal{H}_V$

**while**  $l_i$  nicht zulässig **do**

$\lfloor$   $l_i \leftarrow$  linkestes Label aus  $\mathcal{H}_{\text{right}}$ ,  $\mathcal{H}_{\text{int}}$  und  $\mathcal{H}_V$

    füge  $l_i$  zur Beschriftung hinzu

$f_{\text{new}} \leftarrow$  rechte Kante von  $l_i$

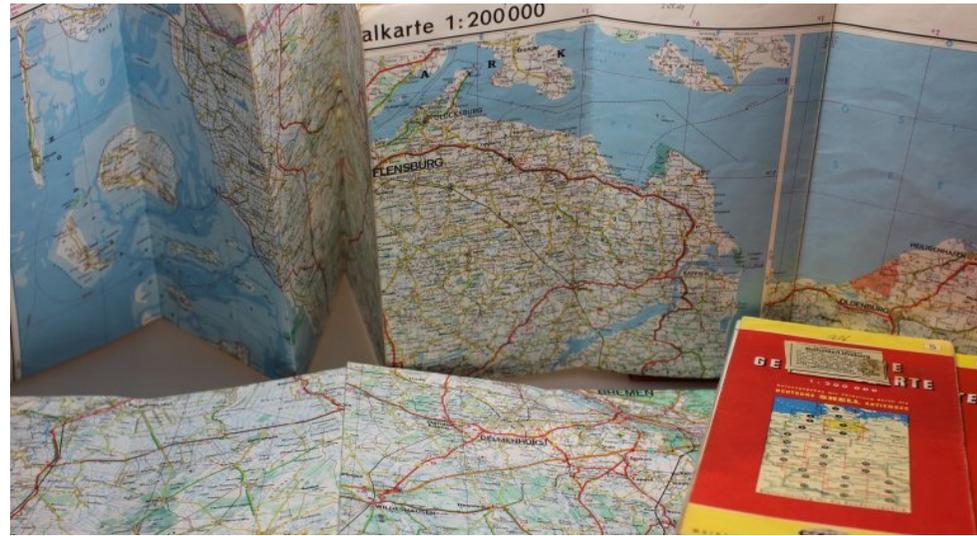
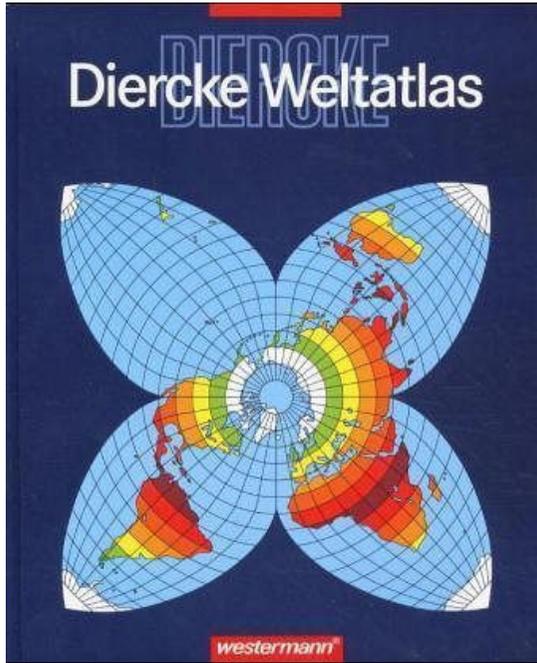
    update  $F$  und  $\mathcal{T}_V$  mit  $f_{\text{new}}$

    suche mit der Region links von  $f_{\text{new}}$  in  $\mathcal{P}_{\text{left}}$  und  $\mathcal{P}_{\text{right}}$  und update

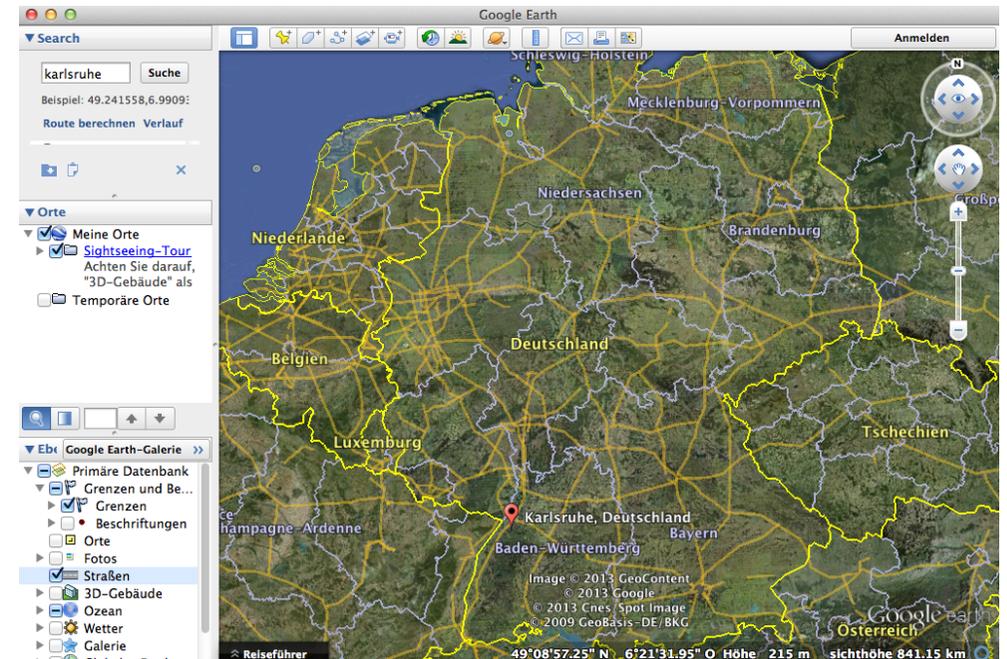
$\mathcal{H}_{\text{right}}$ ,  $\mathcal{P}_{\text{left}}$ ,  $\mathcal{H}_{\text{int}}$ ,  $\mathcal{T}_i$ 's und  $\mathcal{P}_{\text{right}}$  wie beschrieben

    entferne alle Verweise auf  $p_i$  aus den Datenstrukturen

# Was ist eine Landkarte?



oder



# Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.



# Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.



Welche Eigenschaften haben dynamische Karten, wie wirkt sich das auf die Beschriftung aus?

# Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.



Kartenansicht bewegt sich kontinuierlich wenn der Nutzer

- zoomt
- verschiebt
- rotiert
- neigt

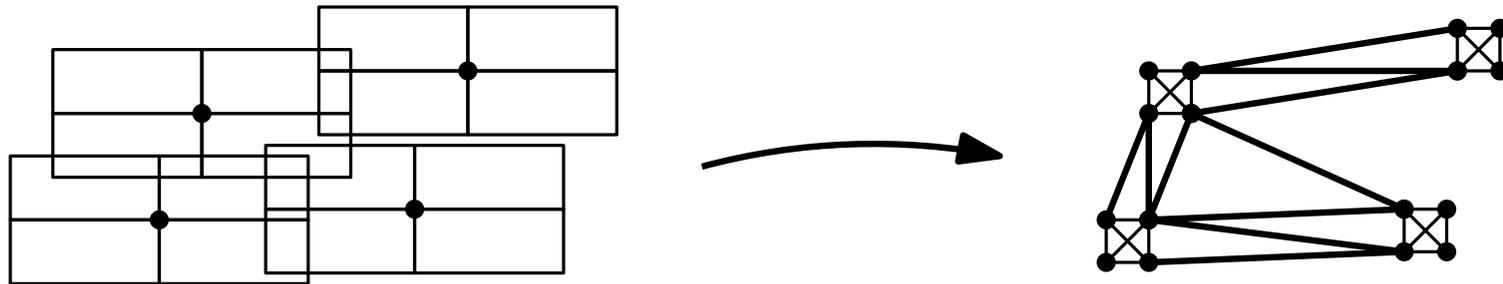
Layout und Beschriftung müssen sich an die dynamische Kartenbewegung anpassen

→ kontinuierliche Generalisierung

→ **kontinuierliche Kartenbeschriftung**

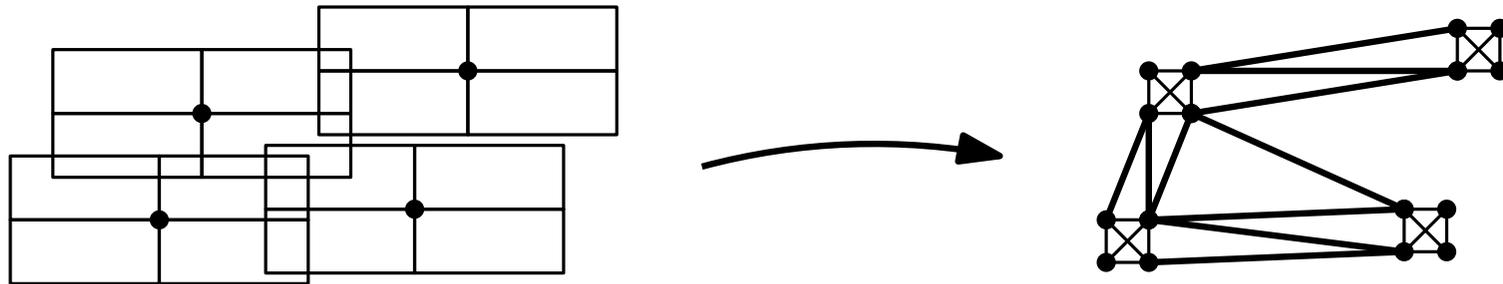
# Lassen sich statische Verfahren nutzen?

Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.



# Lassen sich statische Verfahren nutzen?

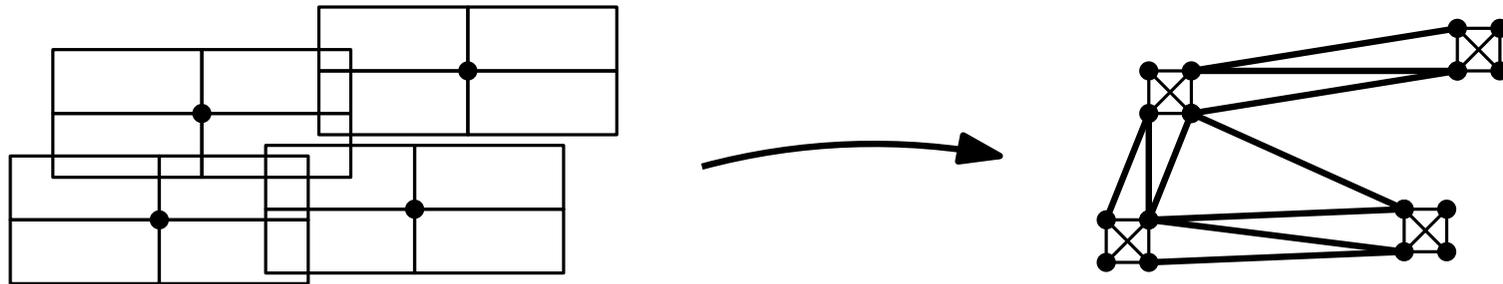
Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.



Übertragung auf **dynamische** Karten: schnelle Algorithmen um jeden Frame einer Animation in Echtzeit zu beschriften.

# Lassen sich statische Verfahren nutzen?

Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.



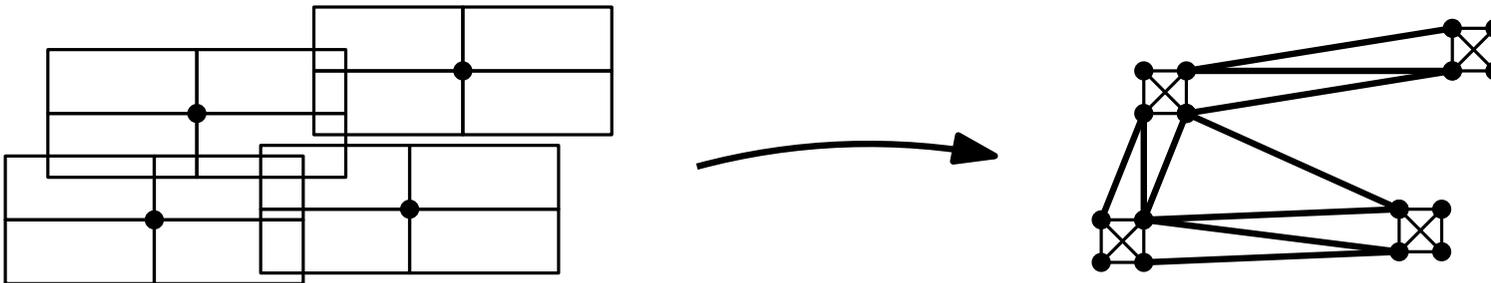
Übertragung auf **dynamische** Karten: schnelle Algorithmen um jeden Frame einer Animation in Echtzeit zu beschriften.

**Ansatz 1:** Vorbereitung & Abfragen [Petzold, Gröger, Plümer '03]

- konstruiere *reaktiven* Konfliktgraph, Abfrage während Interaktion, verwende greedy Verfahren für statische Labelauswahl

# Lassen sich statische Verfahren nutzen?

Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.



Übertragung auf **dynamische** Karten: schnelle Algorithmen um jeden Frame einer Animation in Echtzeit zu beschriften.

**Ansatz 1:** Vorberechnung & Abfragen [Petzold, Gröger, Plümer '03]

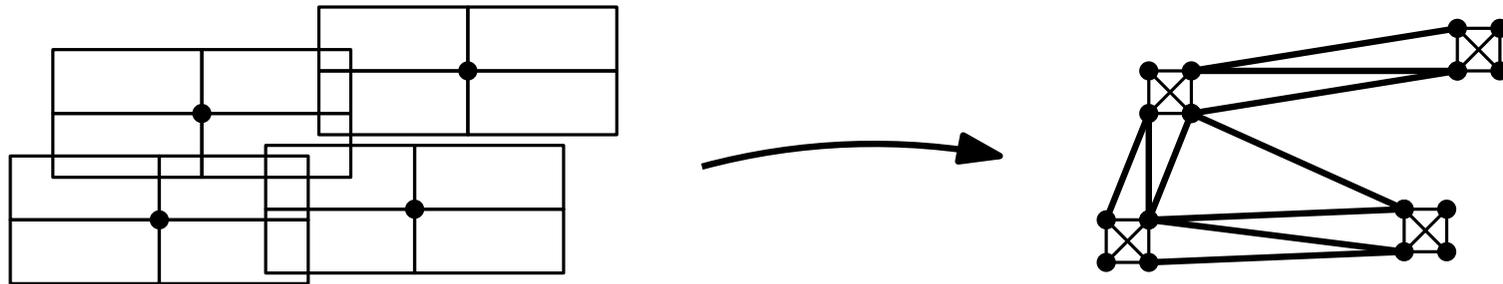
- konstruiere *reaktiven* Konfliktgraph, Abfrage während Interaktion, verwende greedy Verfahren für statische Labelauswahl

**Ansatz 2:** schnelle on-the-fly Berechnung

- berechne lokalen Konfliktgraph, schätze Positionskosten, suche greedy günstigste Position [Mote '07]
- partikelbasierte Konflikterkennung, sequentielle Platzierung nach absteigender Positionspräferenz [Luboschik, Schumann, Cords '08]

# Lassen sich statische Verfahren nutzen?

Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.



Übertragung auf **dynamische** Karten: schnelle Algorithmen um jeden Frame einer Animation in Echtzeit zu beschriften.

**Ansatz 1:** Vorberechnung & Abfragen [Petzold, Gröger, Plümer '03]

- konstruiere *reaktiven* Konfliktgraph, Abfrage während Interaktion, verwende greedy Verfahren für statische Labelauswahl

**Ist das Problem damit gelöst?**

**Ansatz 2:** schnelle *on-the-fly* Berechnung

- berechne lokalen Konfliktgraph, schätze Positionskosten, suche greedy günstigste Position [Mote '07]
- partikelbasierte Konflikterkennung, sequentielle Platzierung nach absteigender Positionspräferenz [Luboschik, Schumann, Cords '08]

# Neue Verfahren sind notwendig

Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

- jeder Frame wird unabhängig von Nachbarframes beschriftet
- Label neigen zu Flackern und Sprüngen
- **zeitliche Kohärenz** oder **Konsistenz** wird nicht betrachtet

# Neue Verfahren sind notwendig

Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

- jeder Frame wird unabhängig von Nachbarframes beschriftet
- Label neigen zu Flackern und Sprüngen
- **zeitliche Kohärenz** oder **Konsistenz** wird nicht betrachtet



Aktuelle reale Beispiele zeigen noch immer negative Effekte!



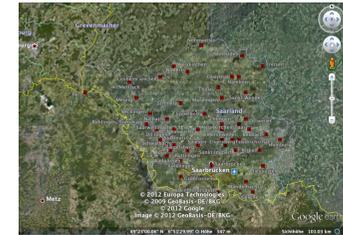
# Neue Verfahren sind notwendig

Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

- jeder Frame wird unabhängig von Nachbarframes beschriftet
- Label neigen zu Flackern und Sprüngen
- **zeitliche Kohärenz** oder **Konsistenz** wird nicht betrachtet



Aktuelle reale Beispiele zeigen noch immer negative Effekte!



## Konsistenzkriterien

[Been, Daiches, Yap '06]

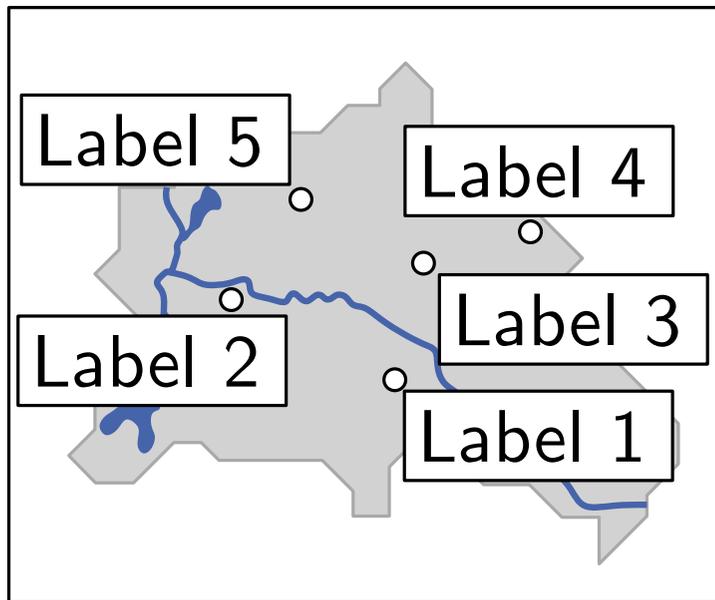
- Label sollen beim Vergrößern nicht verschwinden und beim Verkleinern nicht auftauchen (Monotonität)
- sichtbare Label sollen Position und Größe kontinuierlich ändern
- Platzierung und Auswahl der Label soll eine Funktion des Kartenzustands sein und nicht von der Historie abhängen
- feste Labelgröße auf dem Bildschirm

# 1) Beschriftung für dynamisches Zoomen

[Been et al. 2010]

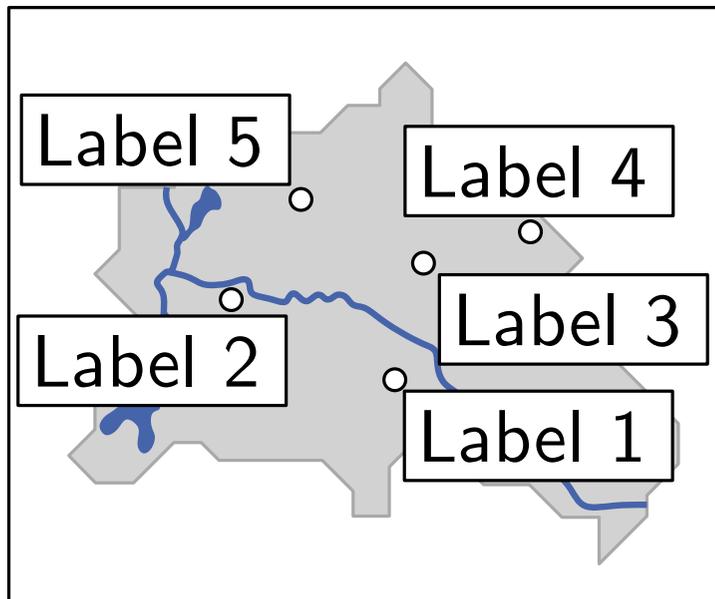
# Label in Karten mit Zoomfunktion

großer Maßstab

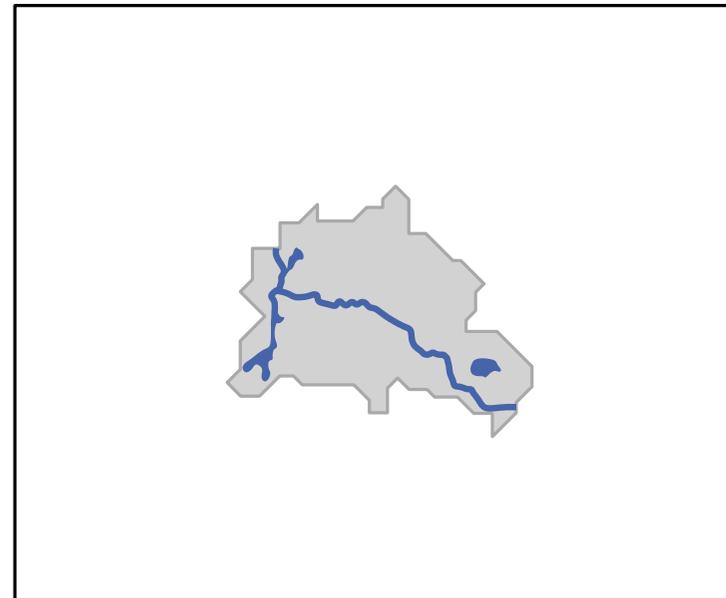


# Label in Karten mit Zoomfunktion

großer Maßstab

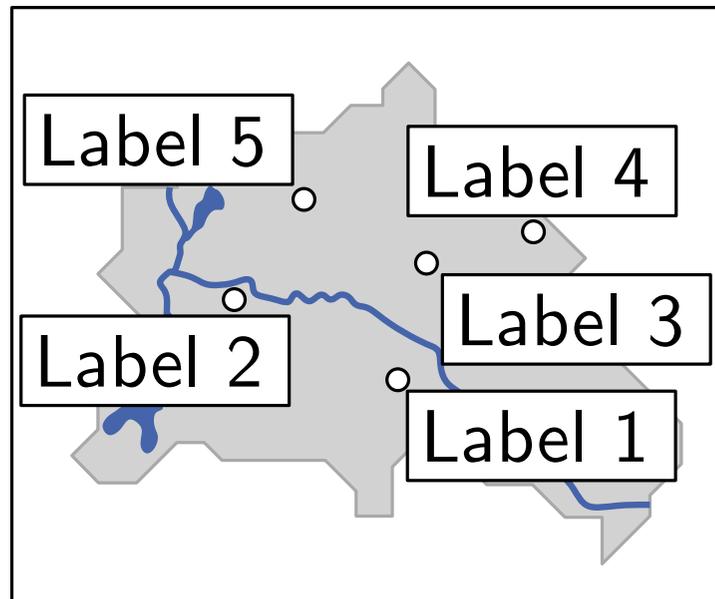


kleiner Maßstab

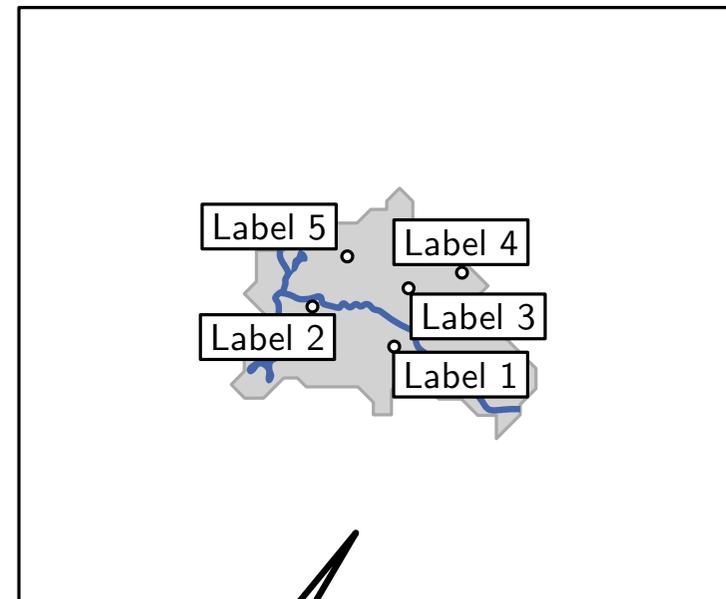


# Label in Karten mit Zoomfunktion

großer Maßstab



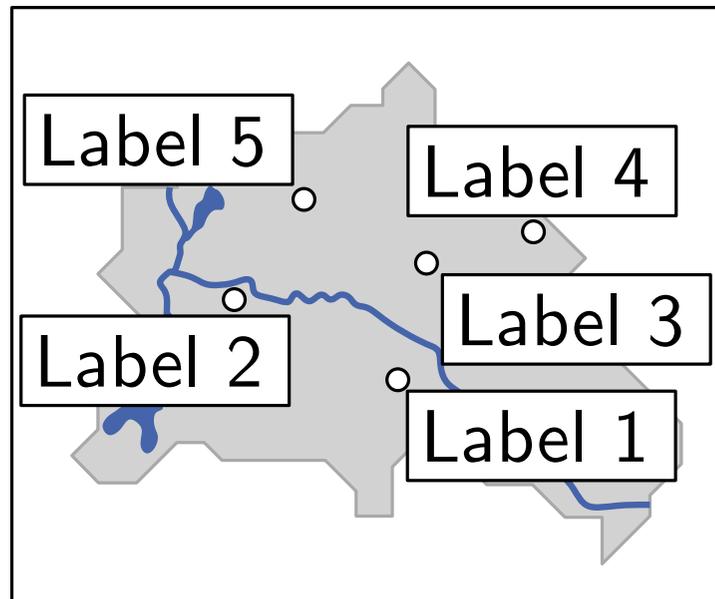
kleiner Maßstab



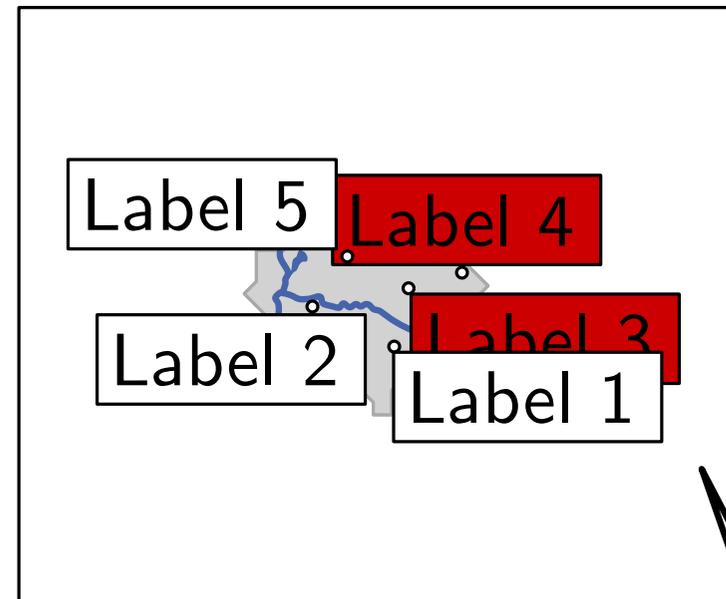
grafisches Zoomen erzeugt kleine Label

# Label in Karten mit Zoomfunktion

großer Maßstab



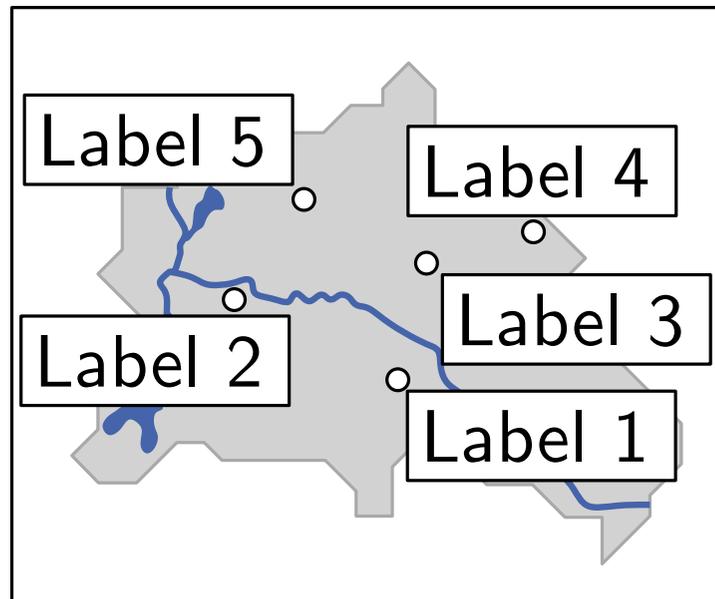
kleiner Maßstab



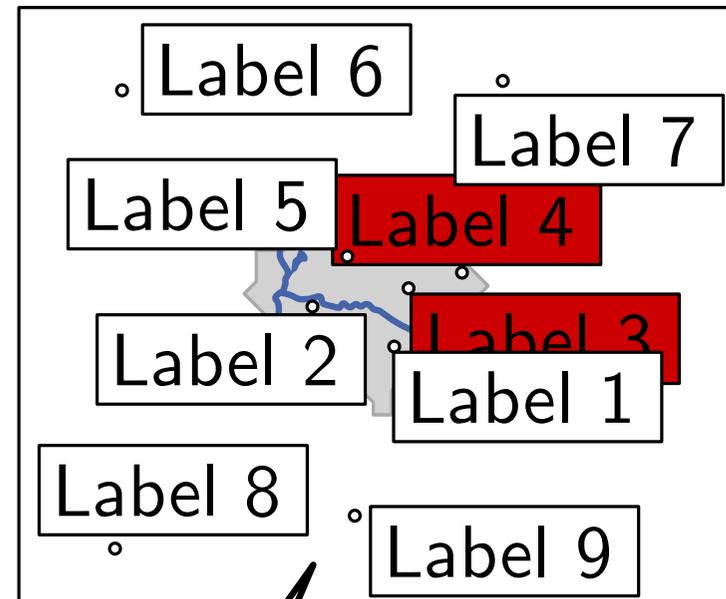
identische Labelgröße erzeugt neue Konflikte

# Label in Karten mit Zoomfunktion

großer Maßstab

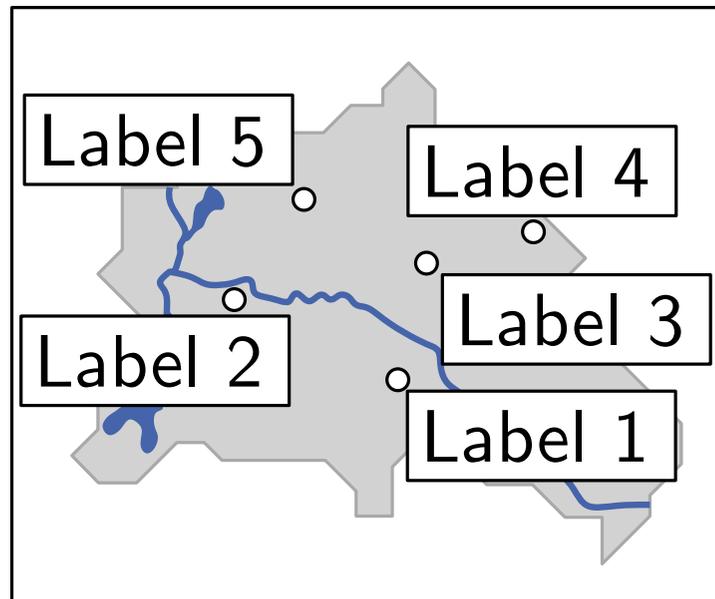


kleiner Maßstab

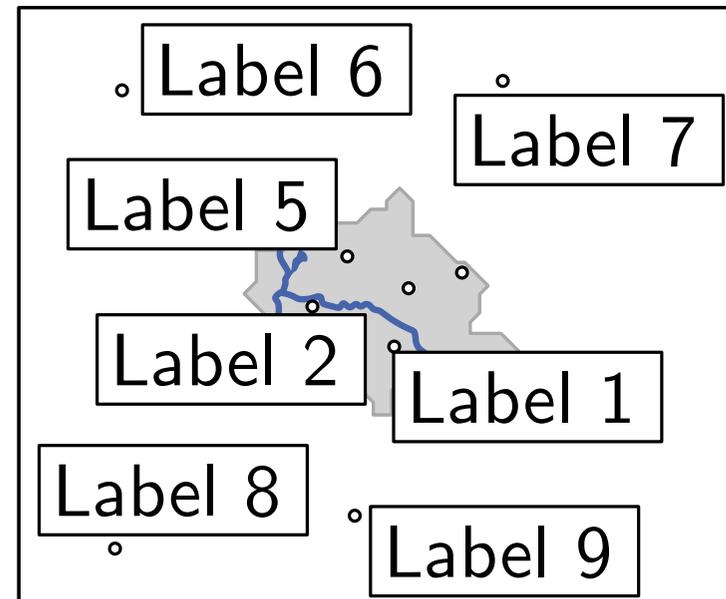


neue Punkte kommen hinzu

großer Maßstab



kleiner Maßstab

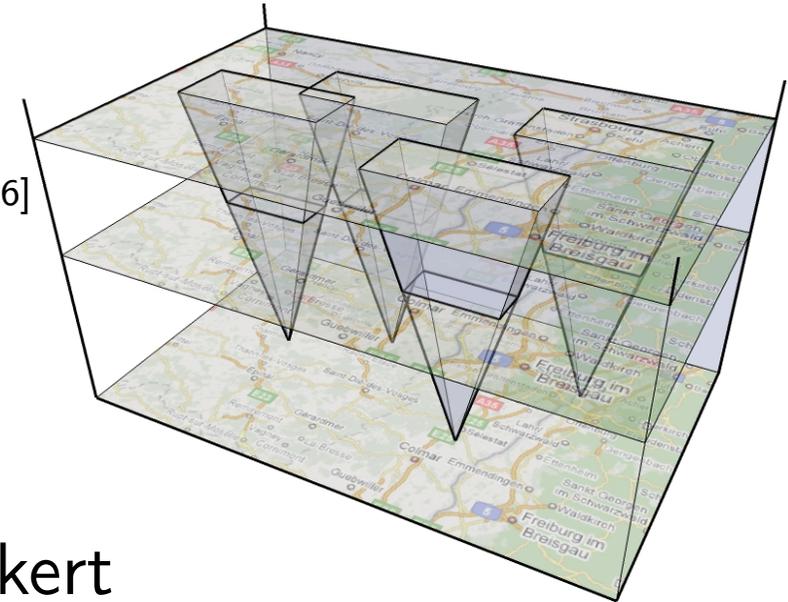


⇒ beim Herauszoomen wachsen Label relativ zur Karte und neue Konflikte müssen aufgelöst werden, ohne dass Label springen oder flackern

## Dynamisches Beschriftungsmodell

[Been, Daiches, Yap '06]

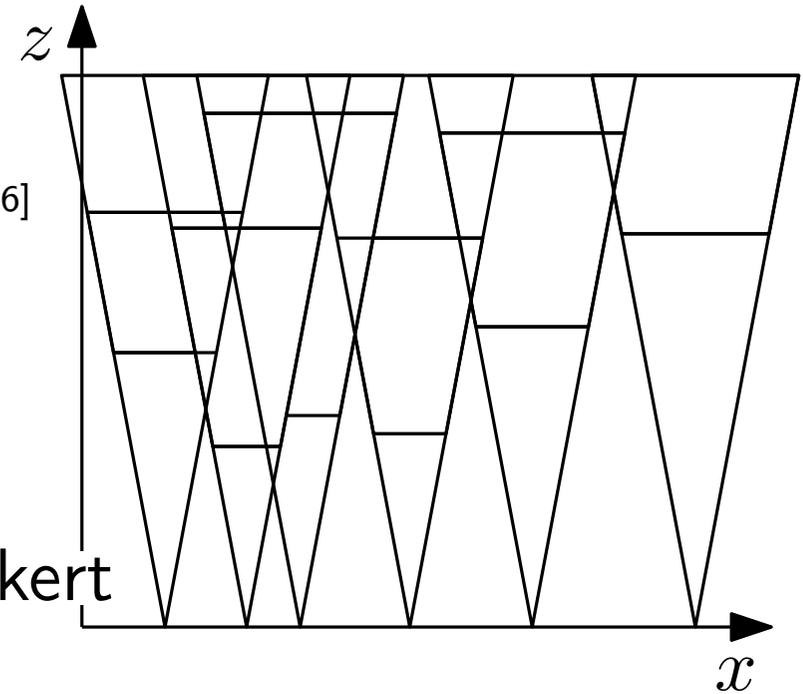
- (inverser) Maßstab auf  $z$ -Achse
- horizontaler Schnitt bei  $z = z_0$   
liefert Karte in Maßstab  $1/z_0$
- jedes Label an festem Punkt verankert  
⇒ **kein Springen**



## Dynamisches Beschriftungsmodell

[Been, Daiches, Yap '06]

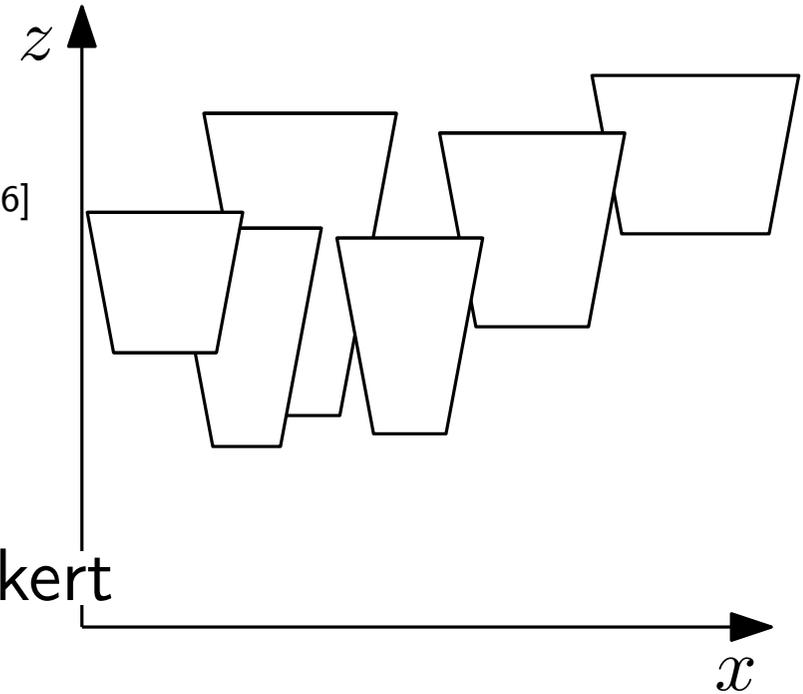
- (inverser) Maßstab auf  $z$ -Achse
- horizontaler Schnitt bei  $z = z_0$   
liefert Karte in Maßstab  $1/z_0$
- jedes Label an festem Punkt verankert  
⇒ **kein Springen**
- verfügbares Maßstabsintervall  $S_L$  für jedes  $L$



## Dynamisches Beschriftungsmodell

[Been, Daiches, Yap '06]

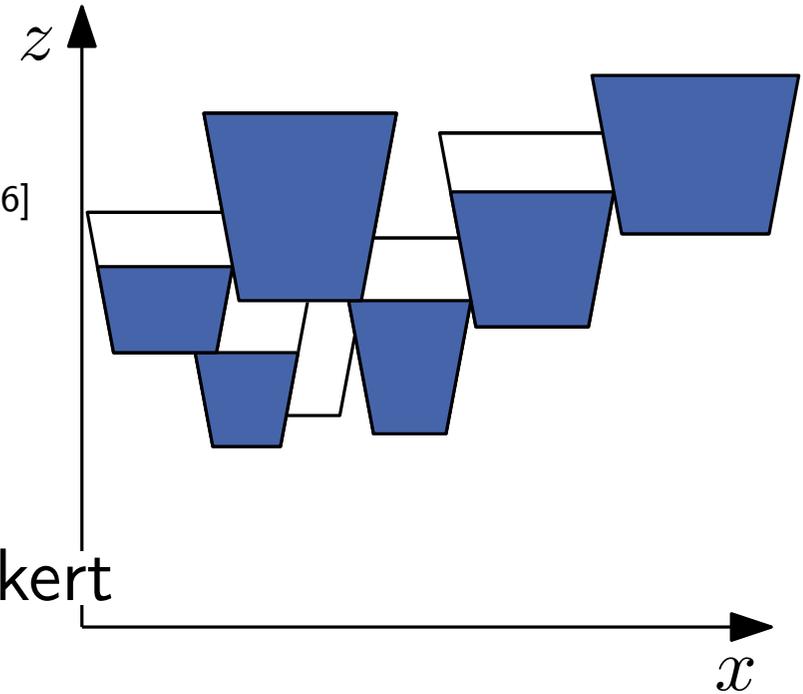
- (inverser) Maßstab auf  $z$ -Achse
- horizontaler Schnitt bei  $z = z_0$   
liefert Karte in Maßstab  $1/z_0$
- jedes Label an festem Punkt verankert  
⇒ **kein Springen**
- verfügbares Maßstabsintervall  $S_L$  für jedes  $L$



## Dynamisches Beschriftungsmodell

[Been, Daiches, Yap '06]

- (inverser) Maßstab auf  $z$ -Achse
- horizontaler Schnitt bei  $z = z_0$  liefert Karte in Maßstab  $1/z_0$
- jedes Label an festem Punkt verankert  
⇒ **kein Springen**
- verfügbares Maßstabsintervall  $S_L$  für jedes  $L$
- berechne ein **aktives Intervall**  $A_L \subseteq S_L$   
⇒ **kein Flackern**
- aktive Intervalle müssen disjunkte Pyramidenstümpfe (bzw. allg. Labelkörper) induzieren ⇒ **gültige Beschriftung**
- **Ziel:** maximiere **aktive Gesamthöhe**  $\sum_L |A_L|$



## Satz 1:

Es ist NP-schwer  $\sum_L |A_L|$  zu maximieren, bzw. für geg.  $K > 0$  ist es NP-vollständig zu entscheiden ob es eine Beschriftung mit  $\sum_L |A_L| \geq K$  gibt.\*

\* selbst wenn  $S_L = [0, z_{\max}]$  für alle Label  $L$ !

## Satz 1:

Es ist NP-schwer  $\sum_L |A_L|$  zu maximieren, bzw. für geg.  $K > 0$  ist es NP-vollständig zu entscheiden ob es eine Beschriftung mit  $\sum_L |A_L| \geq K$  gibt.\*  
\* selbst wenn  $S_L = [0, z_{\max}]$  für alle Label  $L$ !

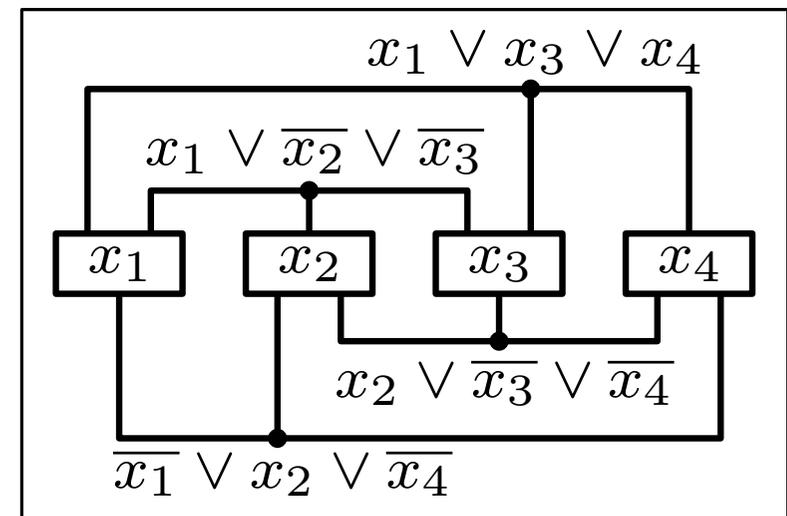
## Beweisskizze:

Reduktion von PLANAREM 3-SAT:  
(NP-schwer: [Lichtenstein '82])

planare 3-Sat Formel  $\varphi$



Menge von Punkten & Labeln,  $K > 0$ :  
 $\varphi$  erfüllbar  $\Leftrightarrow \max \sum_L |A_L| \geq K$



## Satz 1:

Es ist NP-schwer  $\sum_L |A_L|$  zu maximieren, bzw. für geg.  $K > 0$  ist es NP-vollständig zu entscheiden ob es eine Beschriftung mit  $\sum_L |A_L| \geq K$  gibt.\*  
\* selbst wenn  $S_L = [0, z_{\max}]$  für alle Label  $L$ !

## Beweisskizze:

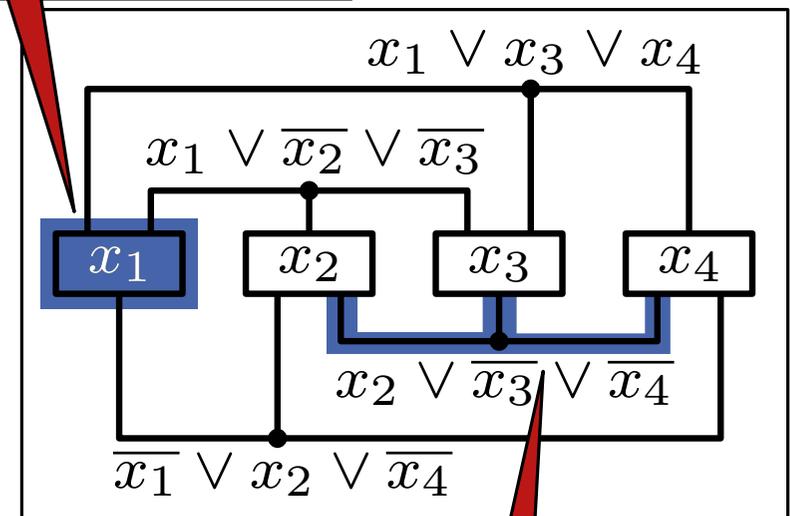
Reduktion von PLANAREM 3-SAT:  
(NP-schwer: [Lichtenstein '82])

planare 3-Sat Formel  $\varphi$



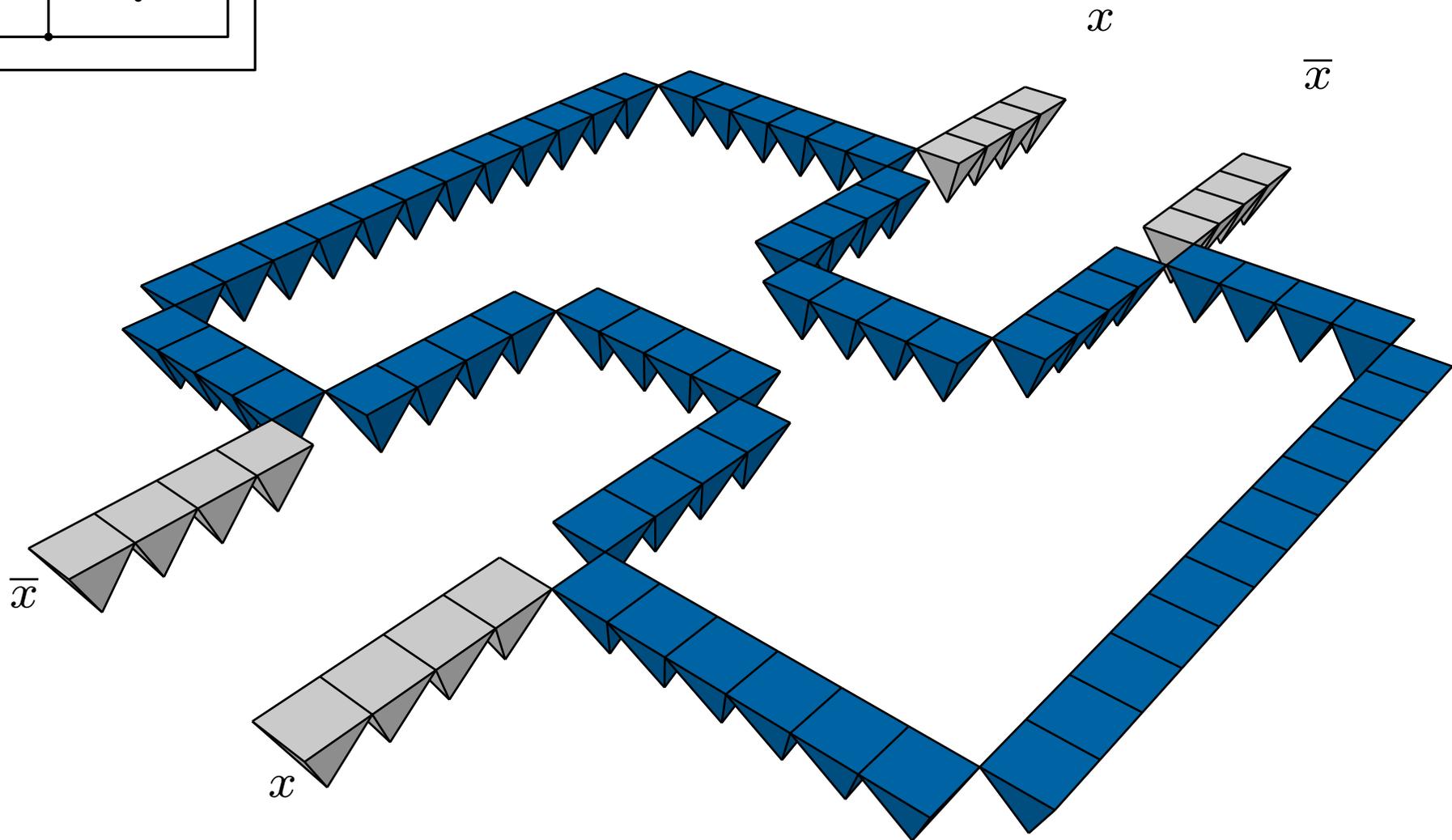
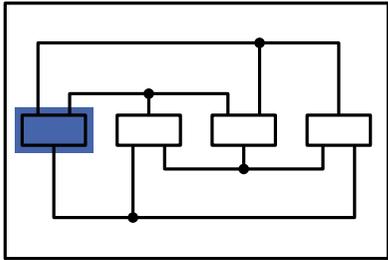
Menge von Punkten & Labeln,  $K > 0$ :  
 $\varphi$  erfüllbar  $\Leftrightarrow \max \sum_L |A_L| \geq K$

Variablengadget



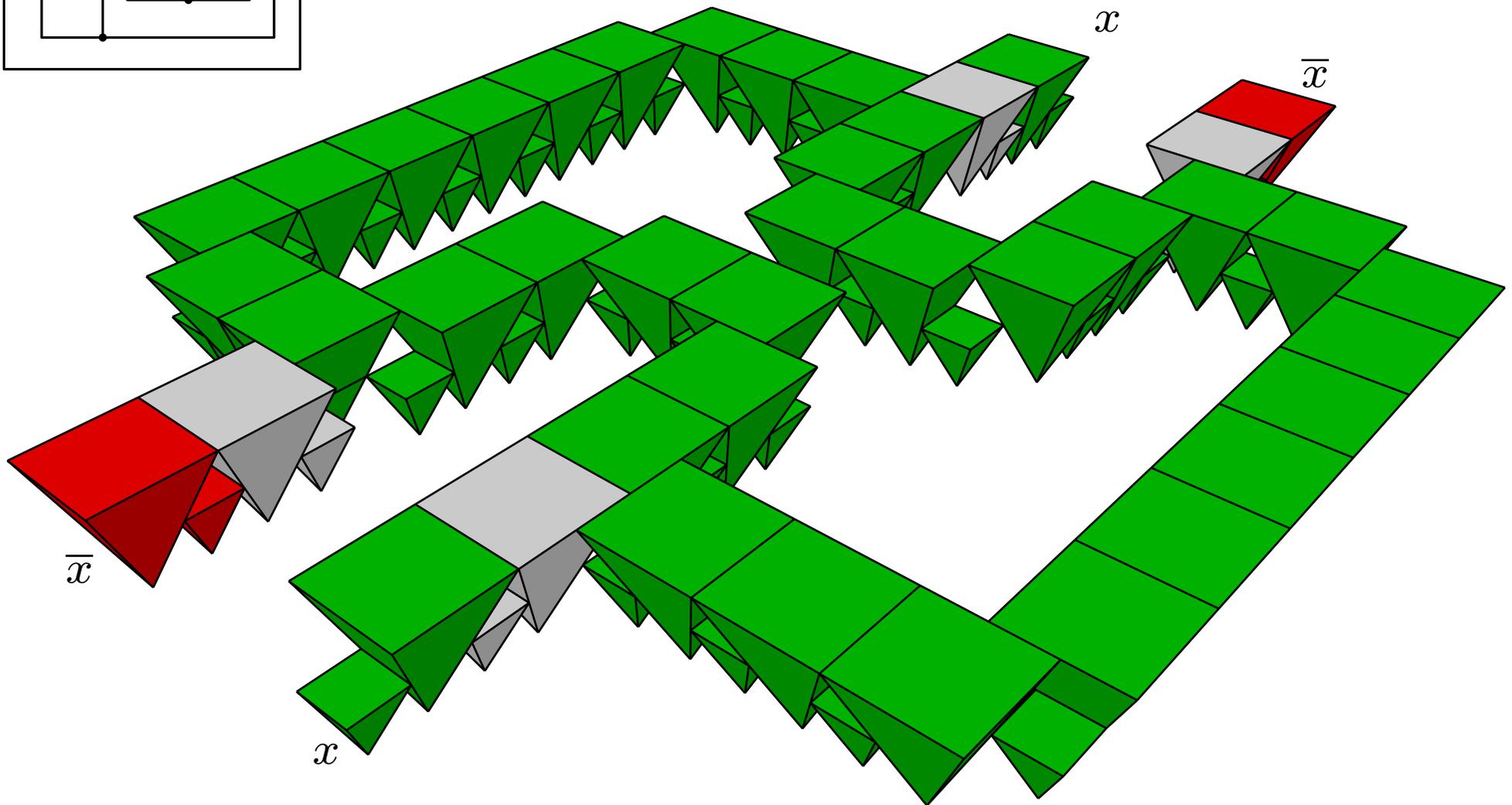
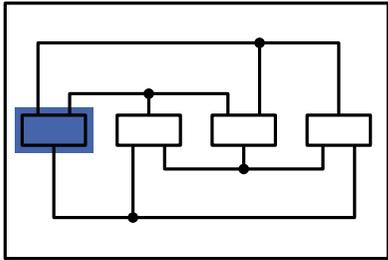
Klauselgadget

# Variablengadget



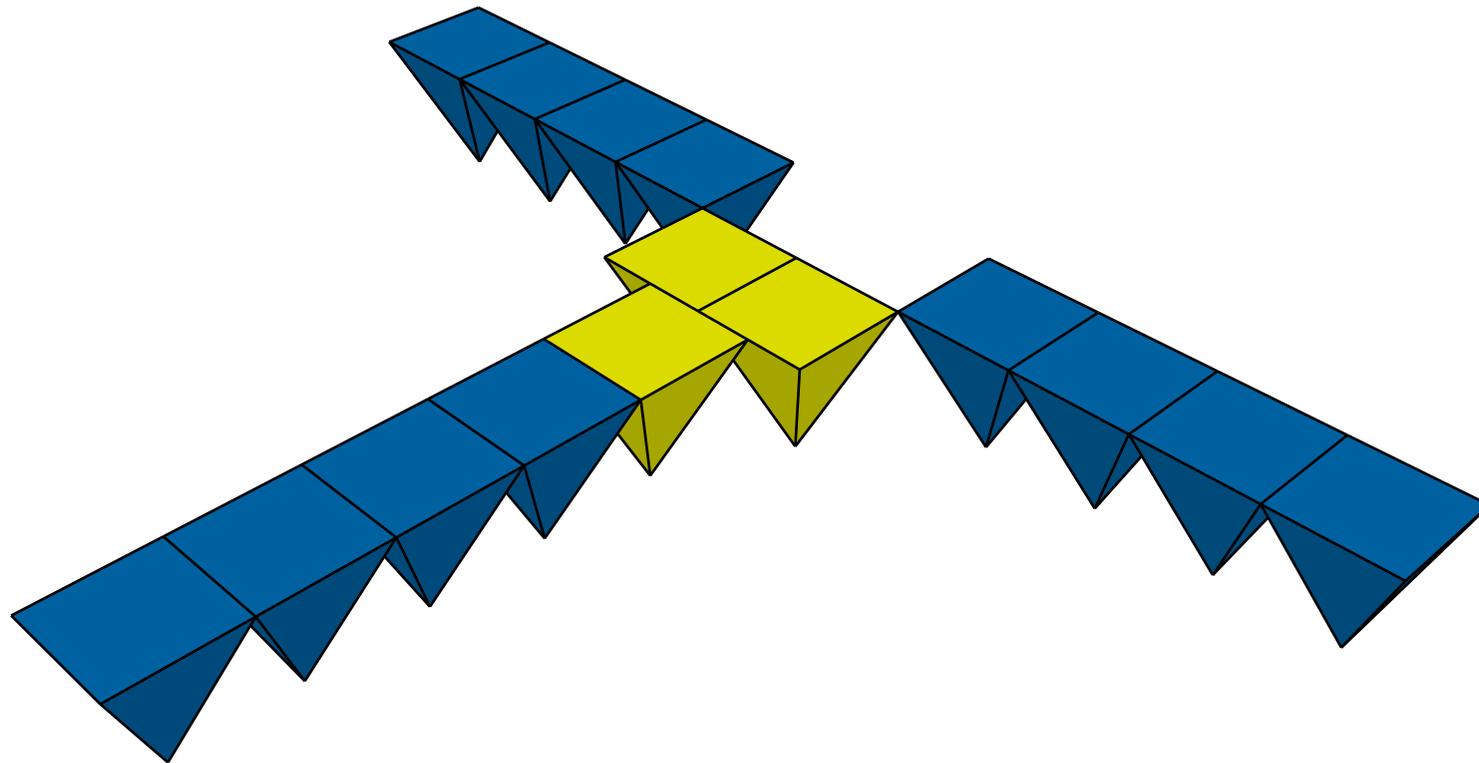
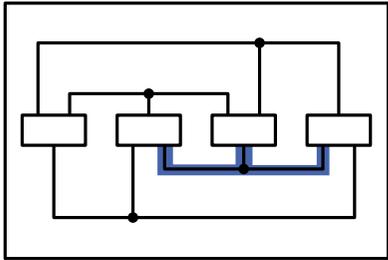
Pyramiden berühren sich bei Höhe  $z_{\max}/2$

# Variablengadget



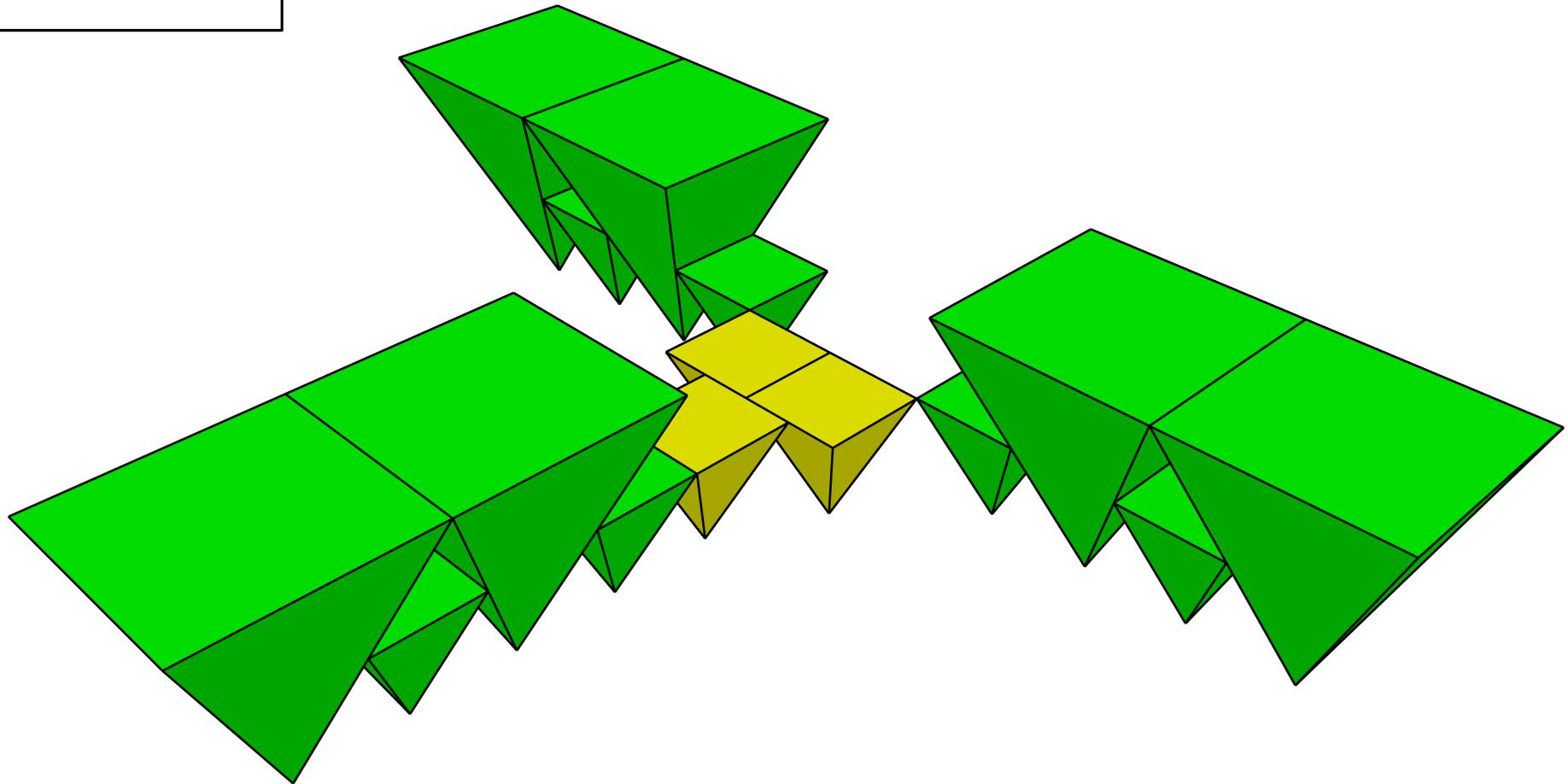
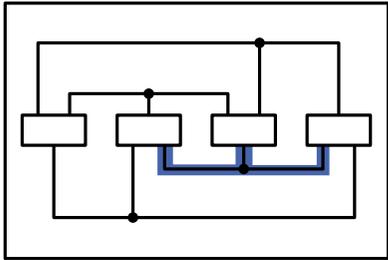
Variable  $x$  ist **wahr**



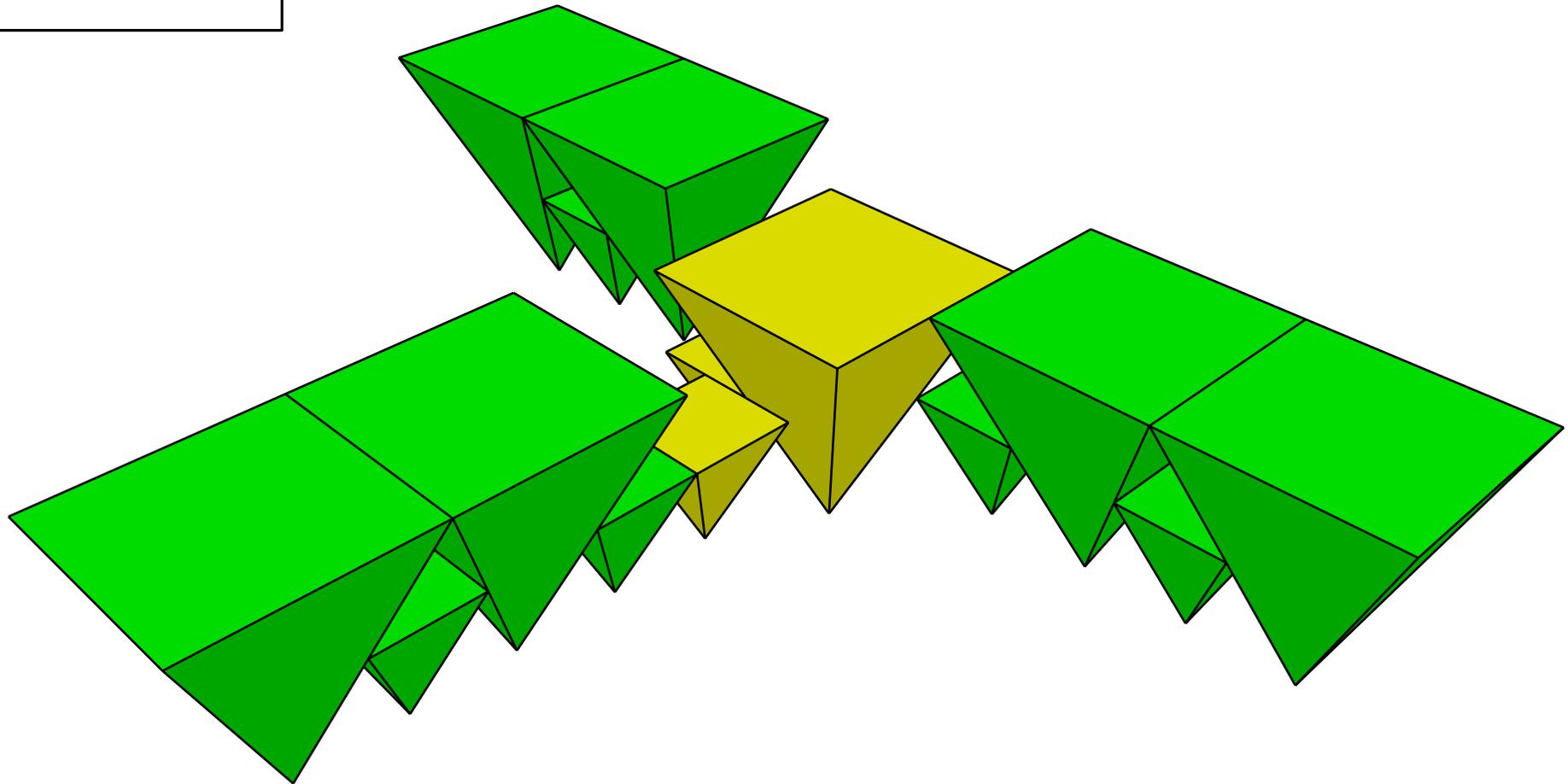
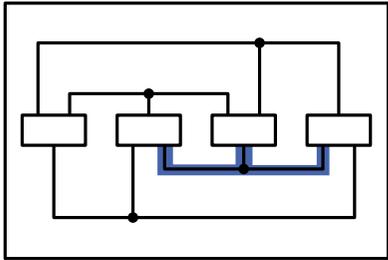


Pyramiden berühren sich bei Höhe  $z_{\max}/2$

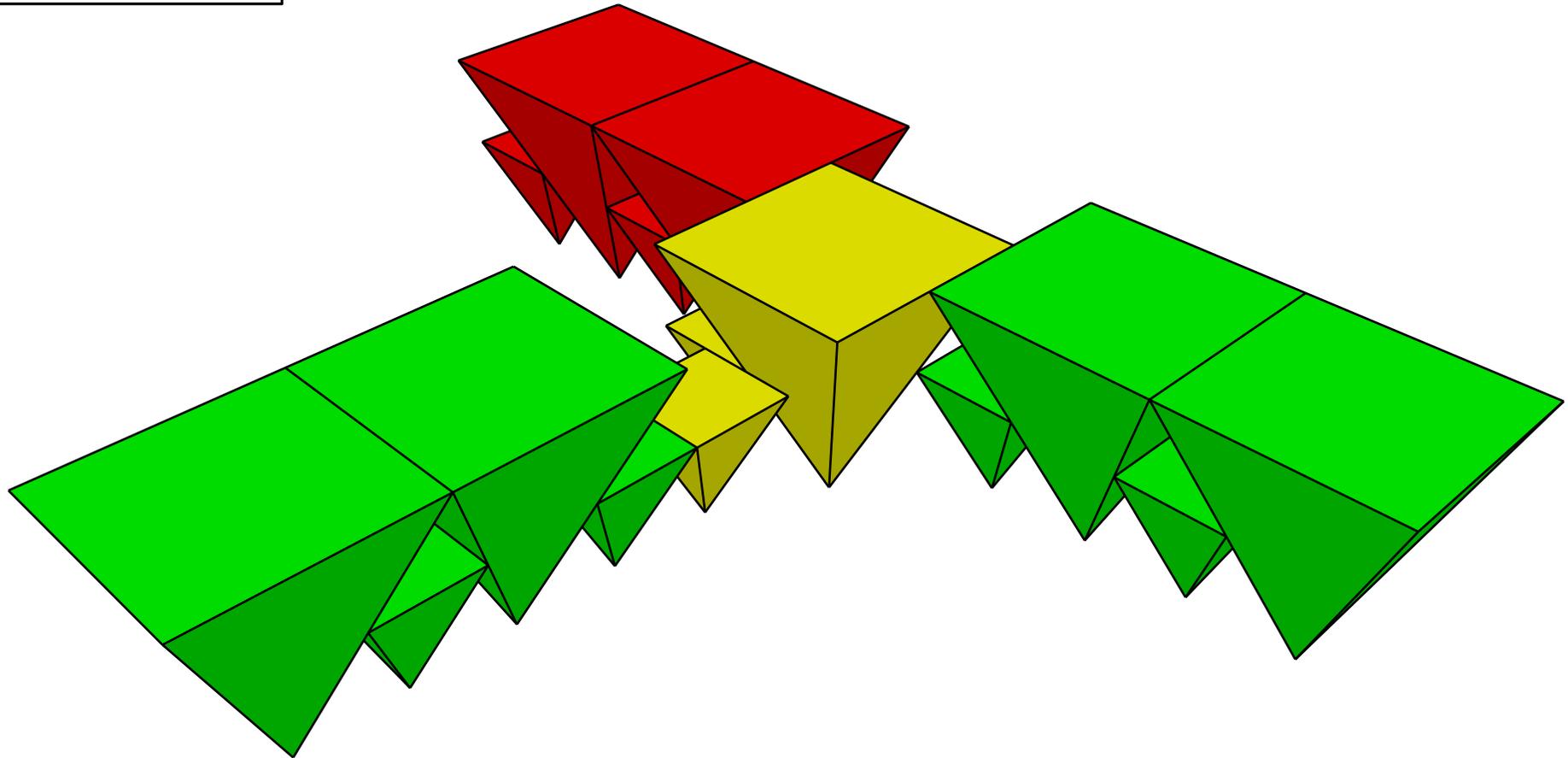
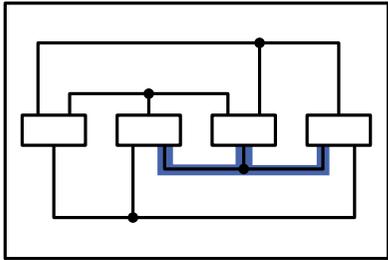
# Klauselgadget



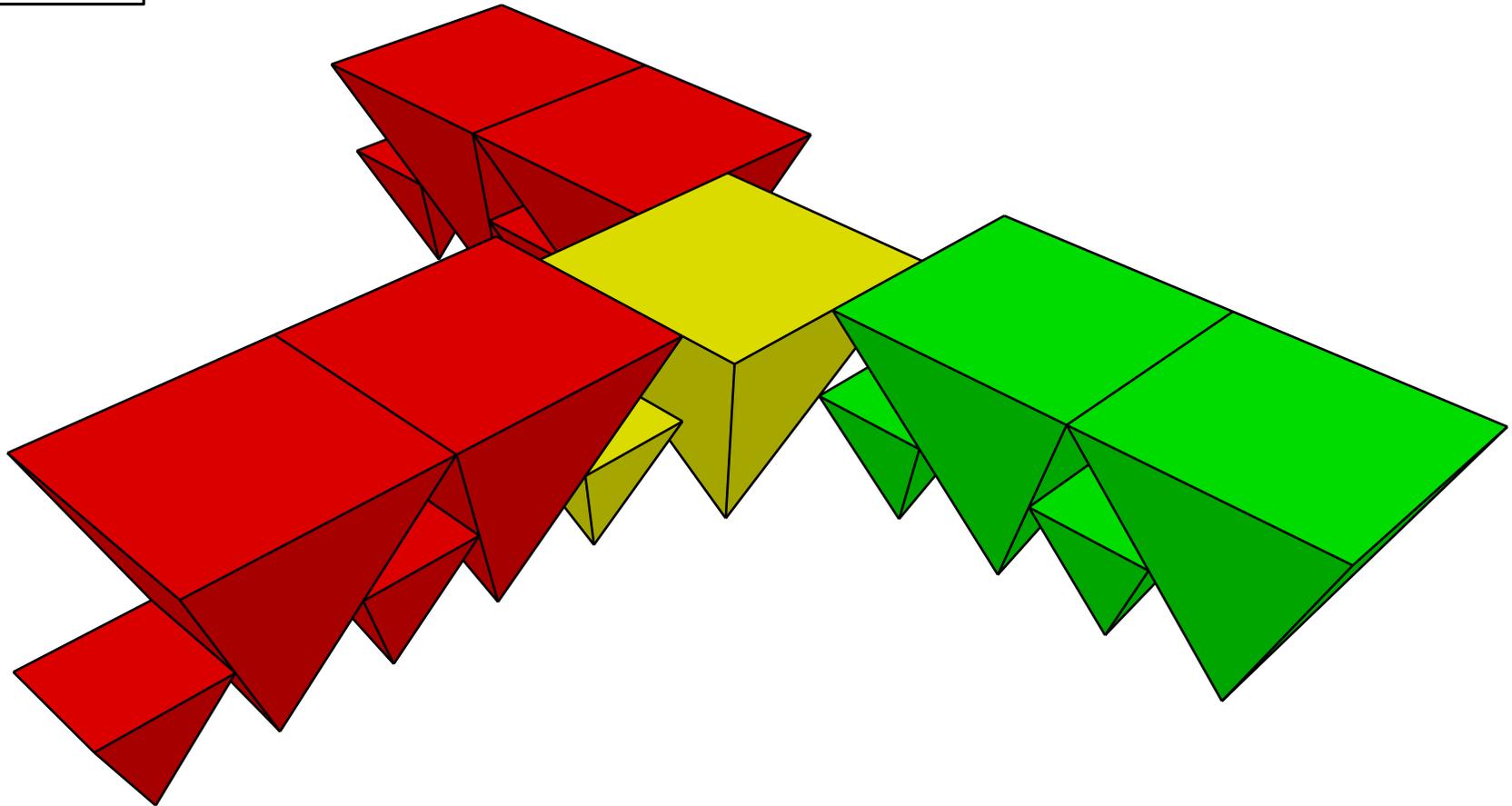
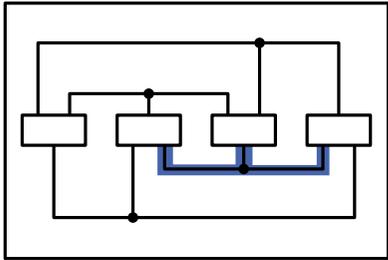
3 Literale sind wahr  $\rightarrow$  Klausel-Beitrag: ?



3 Literale sind wahr  $\rightarrow$  Klausel-Beitrag:  $2 \cdot z_{\max}$

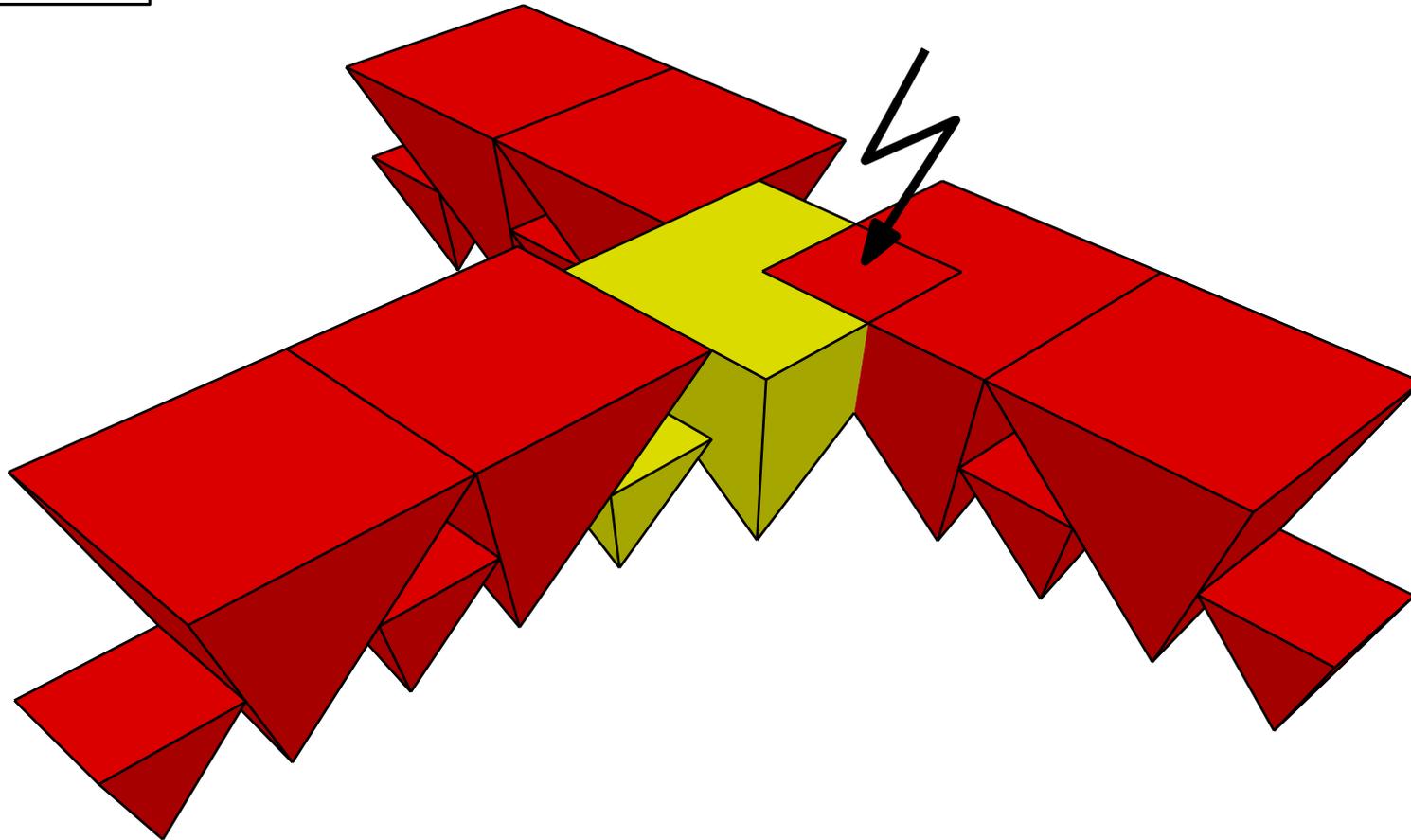
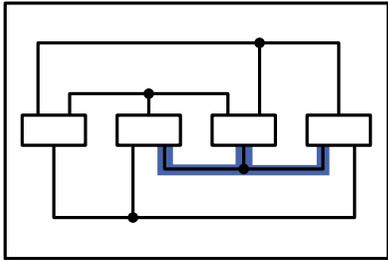


2 Literale sind wahr  $\rightarrow$  Klausel-Beitrag:  $2 \cdot z_{\max}$



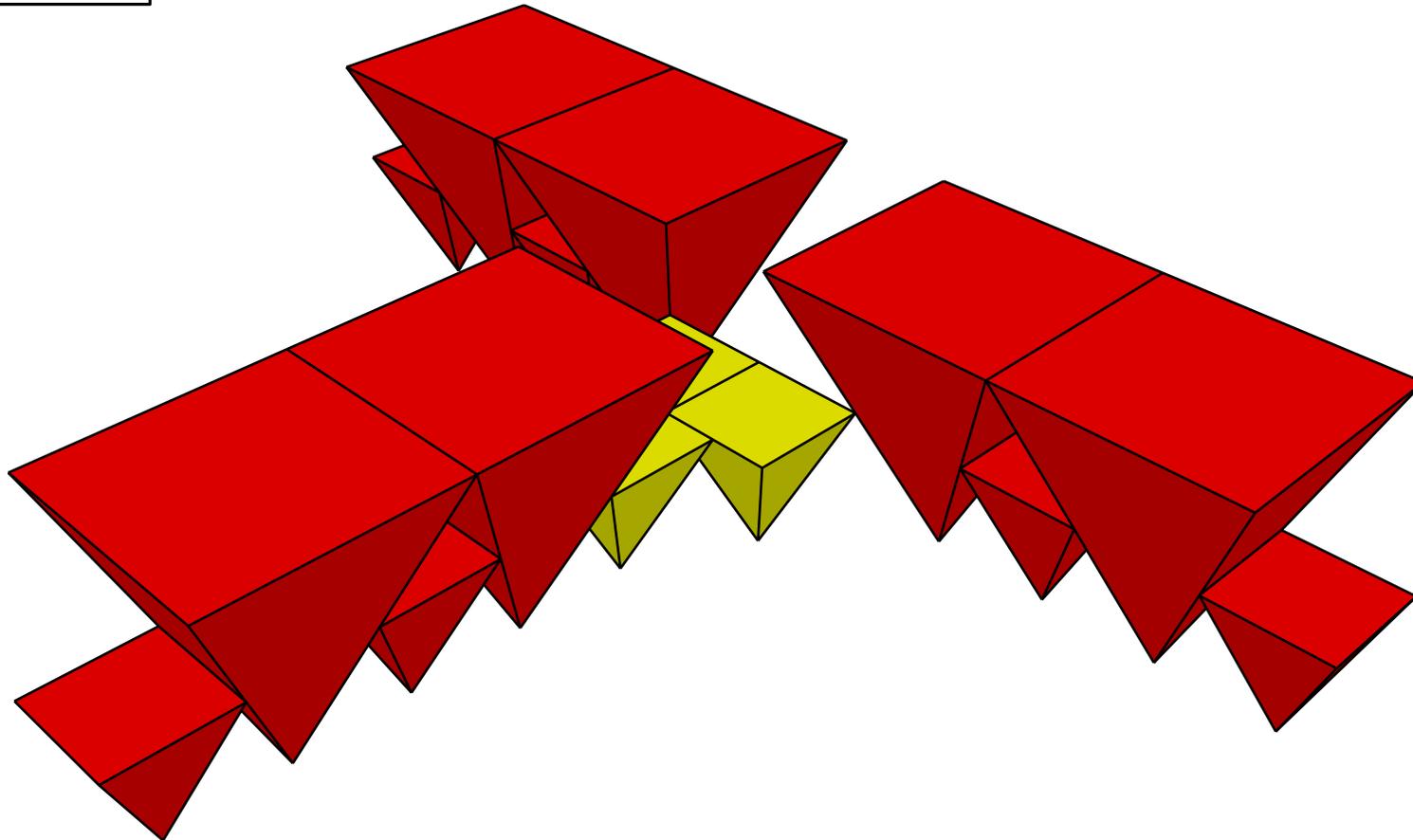
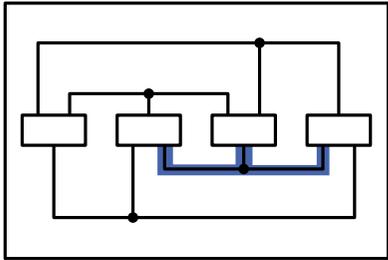
1 Literal ist wahr  $\rightarrow$  Klausel-Beitrag:  $2 \cdot z_{\max}$

# Klauselgadget



0 Literale sind wahr  $\rightarrow$  Klausel-Beitrag: ?

# Klauselgadget



0 Literale sind wahr  $\rightarrow$  Klausel-Beitrag:  $1.5 \cdot z_{\max}$



# Approximationsalgorithmen

# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

$E \leftarrow Q.\text{extractMax}$

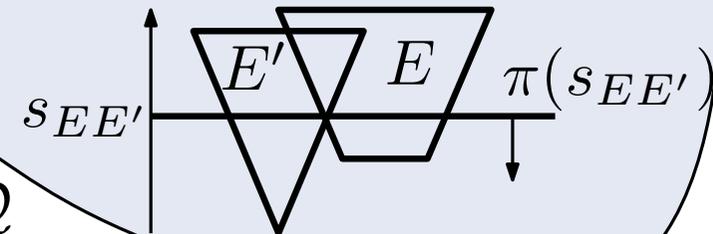
**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$

$s_{EE'}$ : größter  $z$ -Wert, so dass  
 $E \cap E' \cap \pi(s_{EE'}) = \emptyset$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

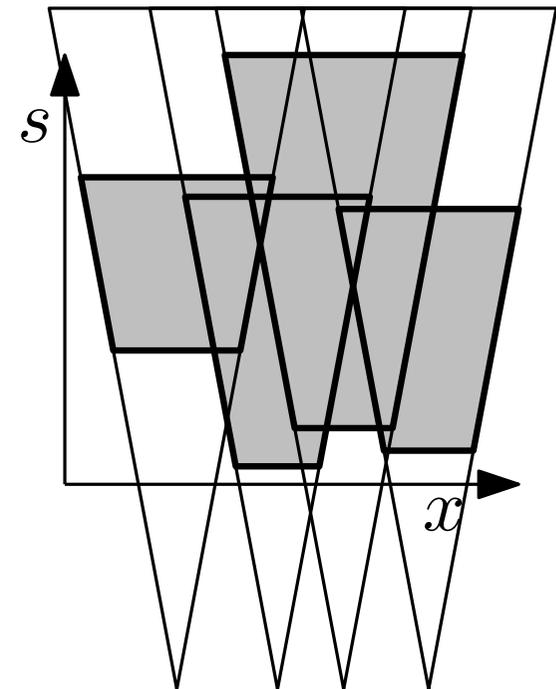
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

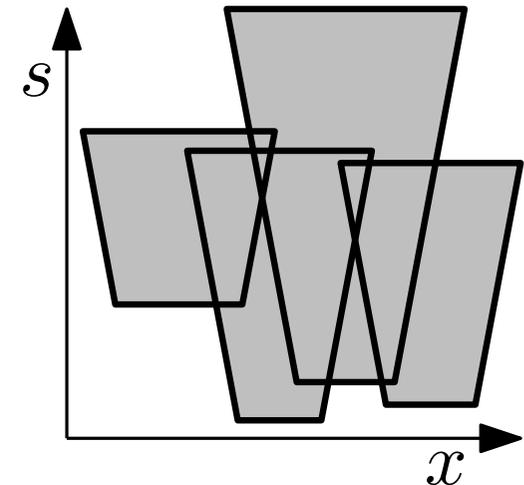
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

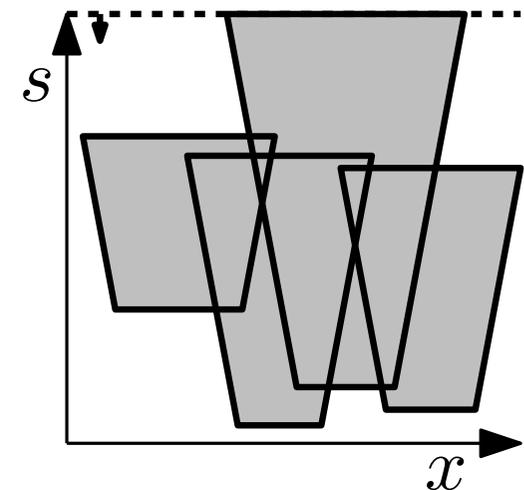
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

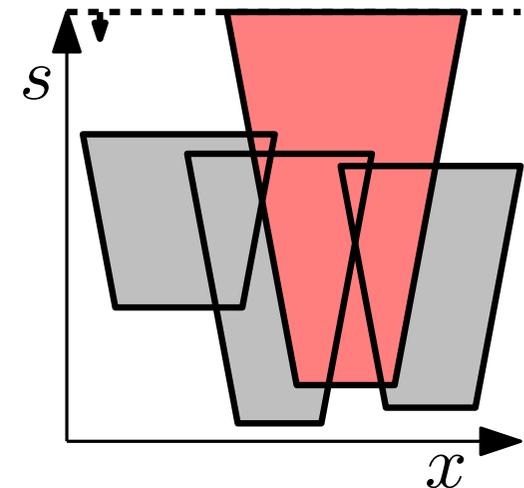
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

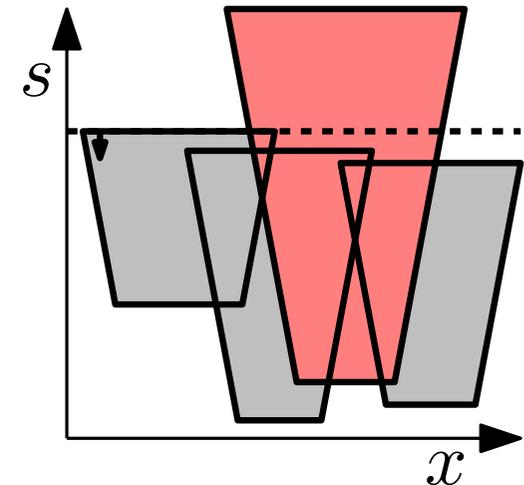
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

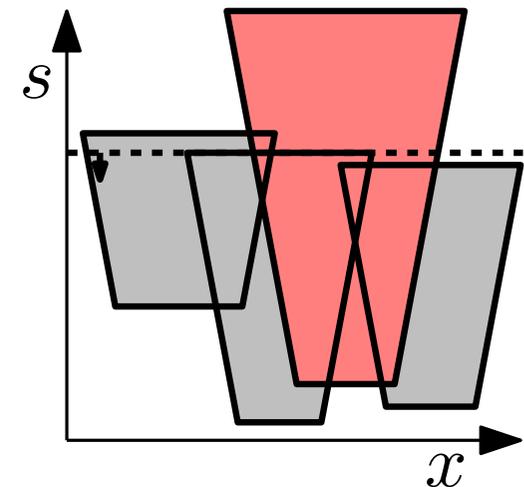
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

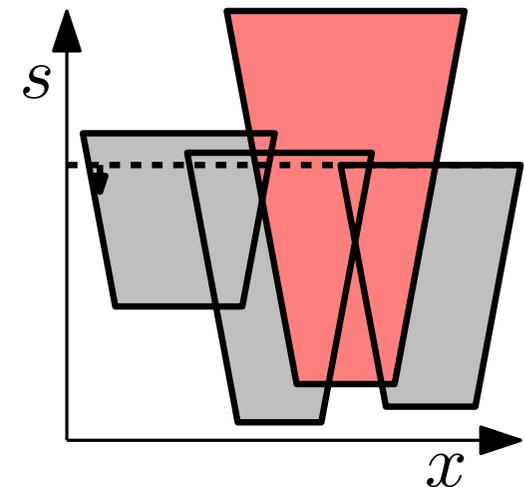
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

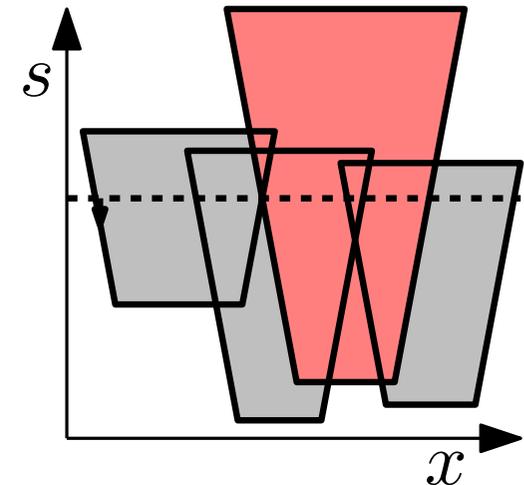
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

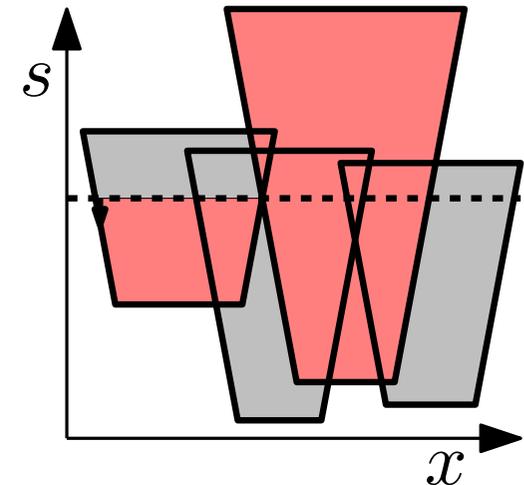
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

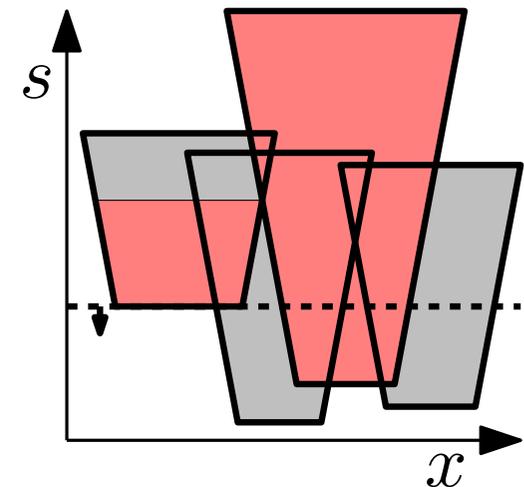
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

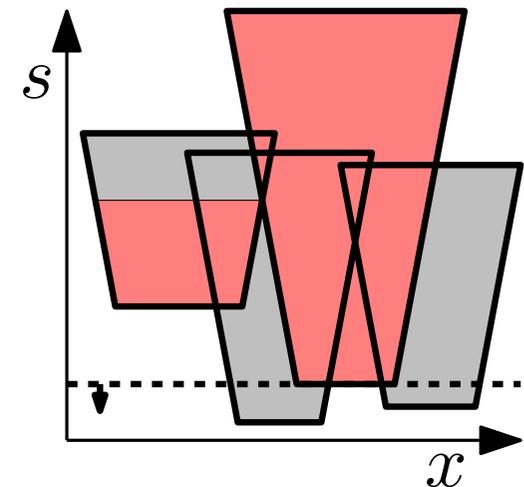
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

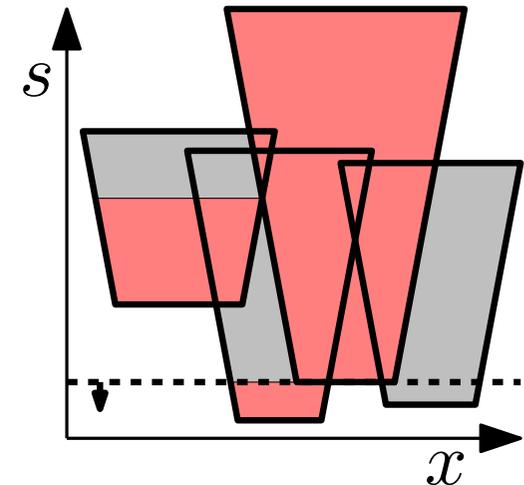
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

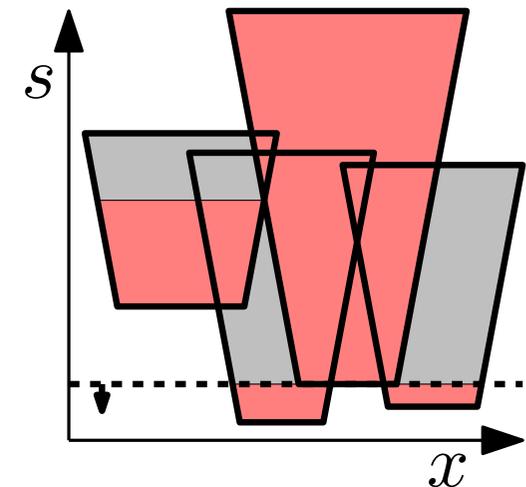
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

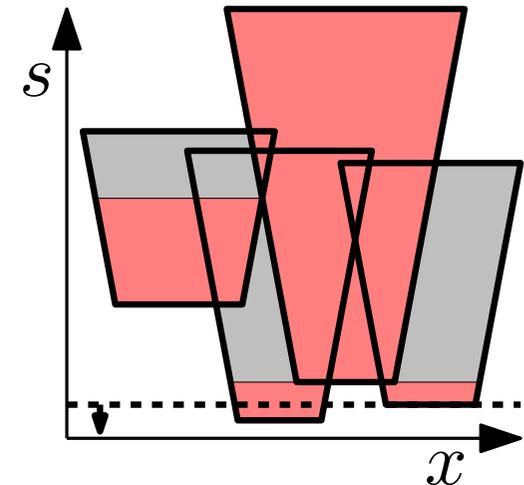
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

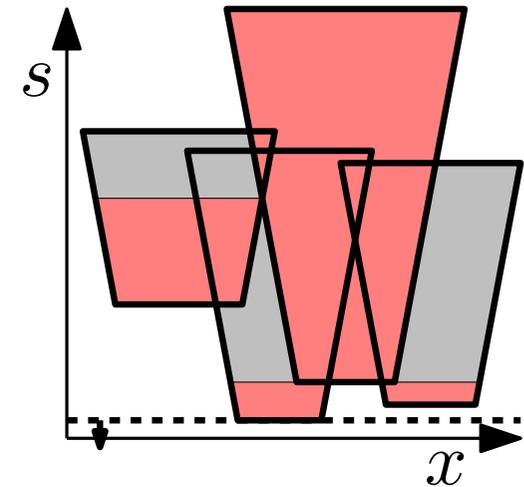
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

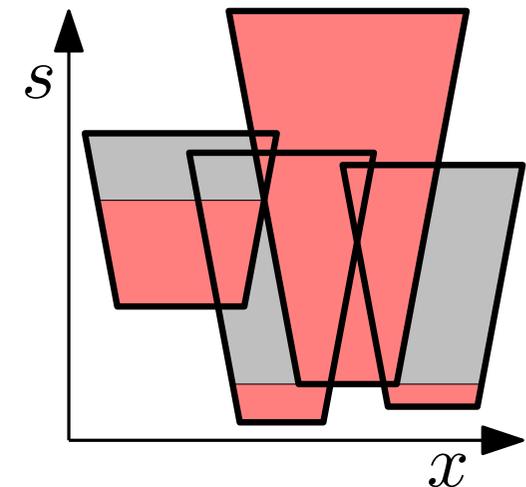
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



Lösung Algorithmus

# Greedy Top-Down Sweep Algorithmus

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

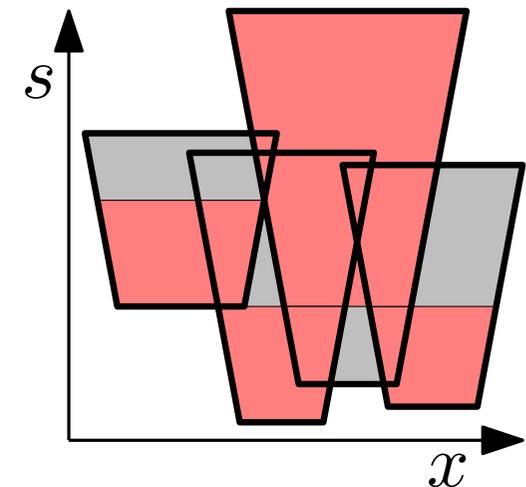
$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$



optimale Lösung

## Algorithmus 1

**Input:** Menge  $\mathcal{E}$  von (offenen) Labelpyramiden, verfügbare Intervalle  $(s_E, S_E)$  für alle  $E \in \mathcal{E}$

**Output:** aktive Intervalle  $(a_E, A_E)$  für alle  $E \in \mathcal{E}$

**foreach**  $E \in \mathcal{E}$  **do**  $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$  Priority Queue für  $\mathcal{E}$  lexikogr. absteigend sortiert nach  $(A_E, S_E)$

**while**  $Q \neq \emptyset$  **do**

$E \leftarrow Q.\text{extractMax}$

**foreach**  $E' \in Q$  **do**

**if**  $E' \cap E \neq \emptyset$  **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

**if**  $A_{E'} = a_{E'}$  **then** lösche  $E'$  aus  $Q$

**Lemma 1:** Algorithmus 1 berechnet eine gültige Beschriftung und lässt sich naiv in  $O(n^2)$  Zeit und  $O(n)$  Platz implementieren.

**Lemma 2:** Sei  $\mathcal{E}$  eine Menge von *Labelkörpern*. Falls gilt, dass jedes  $E \in \mathcal{E}$  für jeden  $z$ -Wert  $s$  nicht mehr als  $c$  paarweise unabhängige Labelkörper blockiert, dann liefert Algorithmus 1 eine  $(1/c)$ -Approximation.

**Lemma 2:** Sei  $\mathcal{E}$  eine Menge von *Labelkörpern*. Falls gilt, dass jedes  $E \in \mathcal{E}$  für jeden  $z$ -Wert  $s$  nicht mehr als  $c$  paarweise unabhängige Labelkörper blockiert, dann liefert Algorithmus 1 eine  $(1/c)$ -Approximation.

Begriffe:

- $E \in \mathcal{E}$  **blockiert**  $E' \in \mathcal{E}$  am Wert  $s$  in einer Lösung  $\mathcal{A}$ , falls  $a_E \leq s \leq A_E$  und  $tr_s(E) \cap tr_s(E') \neq \emptyset$
- die **Spur**  $tr_s(E)$  eines Labelkörpers  $E \in \mathcal{E}$  zum Wert  $s$  ist der Schnitt von  $E$  mit der Ebene  $\pi(s) : z = s$
- $E, E' \in \mathcal{E}$  sind **unabhängig** am Wert  $s$ , falls  $tr_s(E) \cap tr_s(E') = \emptyset$

**Lemma 2:** Sei  $\mathcal{E}$  eine Menge von *Labelkörpern*. Falls gilt, dass jedes  $E \in \mathcal{E}$  für jeden  $z$ -Wert  $s$  nicht mehr als  $c$  paarweise unabhängige Labelkörper blockiert, dann liefert Algorithmus 1 eine  $(1/c)$ -Approximation.

Begriffe:

- $E \in \mathcal{E}$  **blockiert**  $E' \in \mathcal{E}$  am Wert  $s$  in einer Lösung  $\mathcal{A}$ , falls  $a_E \leq s \leq A_E$  und  $tr_s(E) \cap tr_s(E') \neq \emptyset$
- die **Spur**  $tr_s(E)$  eines Labelkörpers  $E \in \mathcal{E}$  zum Wert  $s$  ist der Schnitt von  $E$  mit der Ebene  $\pi(s) : z = s$
- $E, E' \in \mathcal{E}$  sind **unabhängig** am Wert  $s$ , falls  $tr_s(E) \cap tr_s(E') = \emptyset$

Was ist  $c$  für den Fall von quadratischen Einheitslabeln?  
Was ist  $c$  für den Fall einheitlich großer Rechtecke?