Karlsruhe Institute of Technology

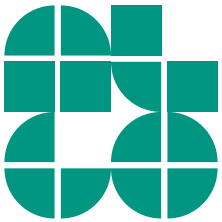
Vorlesung Algorithmische Kartografie

Punktbeschriftung in Landkarten

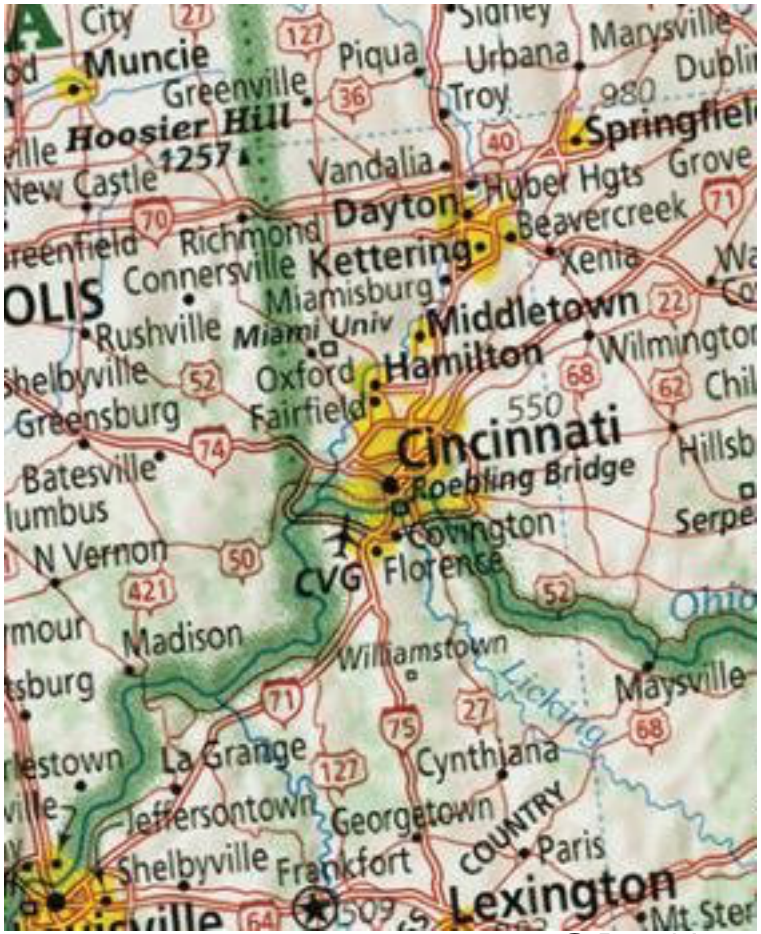
LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Martin Nöllenburg

07.05.2013



Klassische Kartenbeschriftung



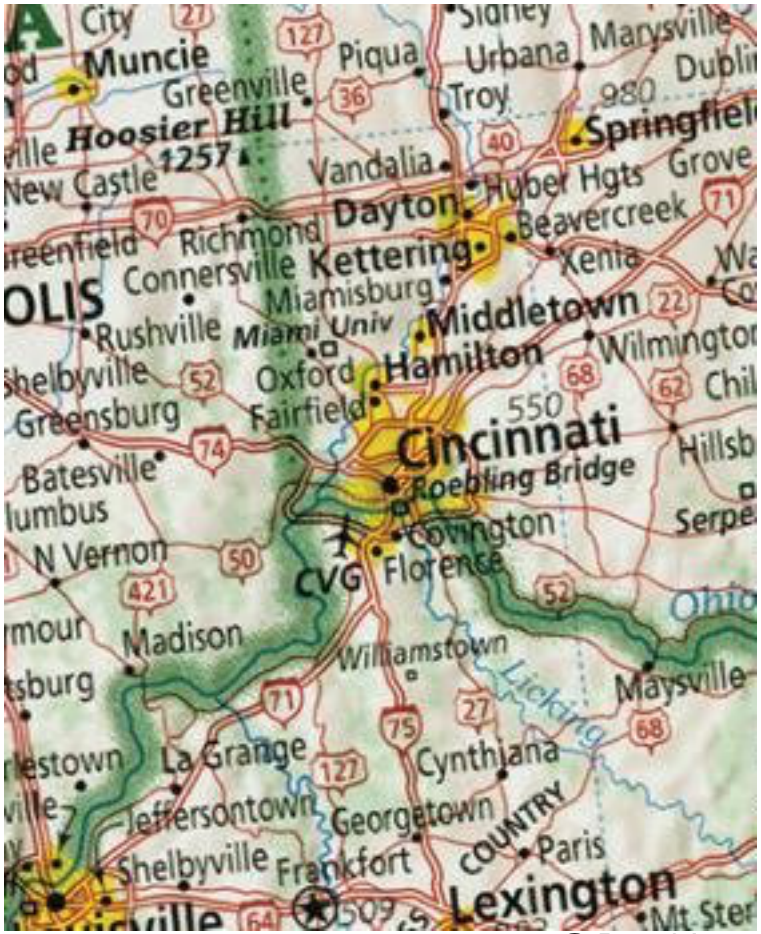
©David Imus

"Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading." (E. Imhof '75)

Kartografie hat lange Erfahrung mit manueller Beschriftung in Karten.

einige Richtlinien:

Klassische Kartenbeschriftung



©David Imus

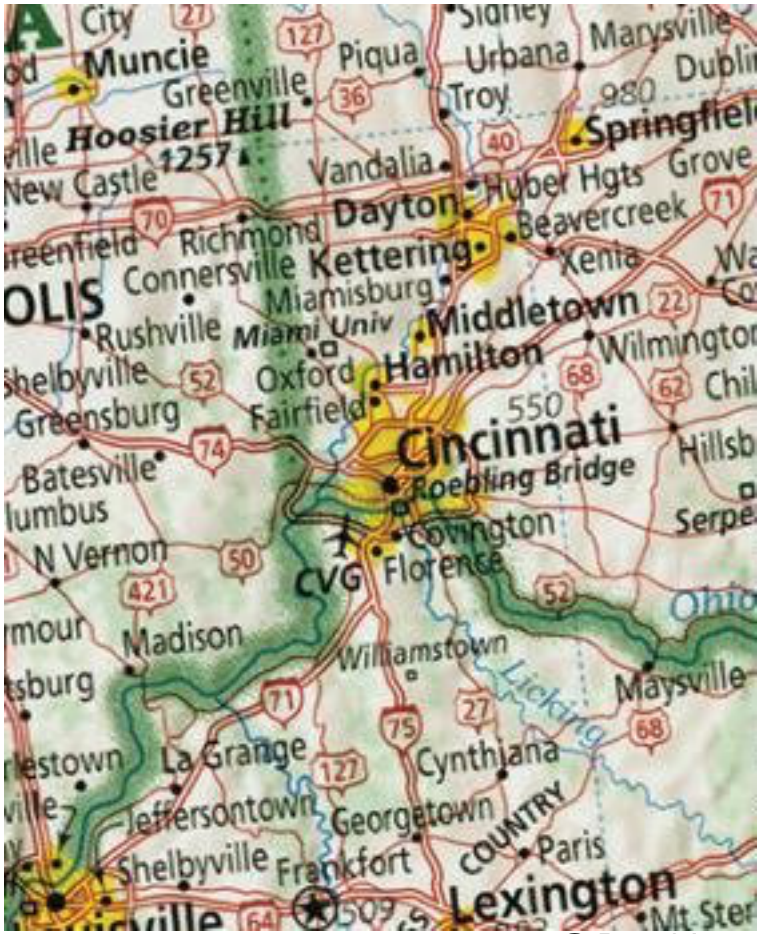
"Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading." (E. Imhof '75)

Kartografie hat lange Erfahrung mit manueller Beschriftung in Karten.

einige Richtlinien:

- neben, über oder unter dem Objekt
- am besten oben rechts
- Überlappungen und Überdeckungen vermeiden
- eindeutige grafische Zuordnung
- ...

Klassische Kartenbeschriftung



©David Imus

"Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading." (E. Imhof '75)

Kartografie hat lange Erfahrung mit manueller Beschriftung in Karten.

einige Richtlinien:

- neben, über oder unter dem Objekt
- am besten oben rechts
- Überlappungen und Überdeckungen vermeiden
- eindeutige grafische Zuordnung
- ...

→ lässt sich leicht in ein Problem der algorithmischen Geometrie zur automatischen Platzierung von Labeln übersetzen

Klassische Kartenbeschriftung

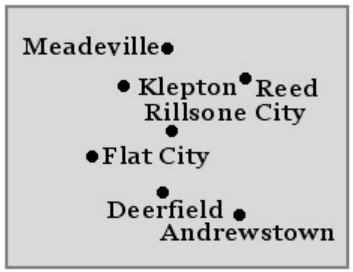
"Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading." (E. Imhof '75)

Viele andere Eigenschaften tragen zur Beschriftungsqualität bei:

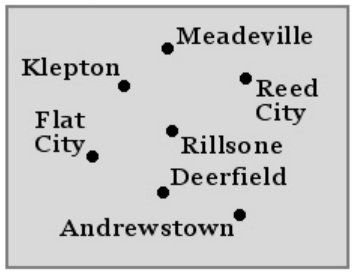
- Font, Farbe
- **fett**/*kursiv* etc.
- Größe, Zeichenabstand

Hier: nur geometrische Platzierung

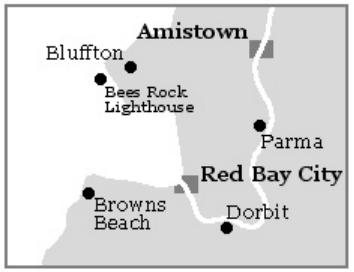
Poor type placement:



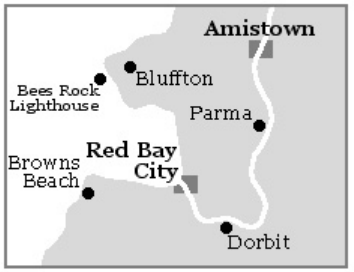
Good type placement:



Poor type placement:



Good type placement:



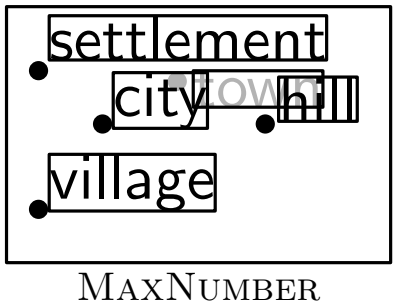
Beispiele von J B Krygier

→ lässt sich leicht in ein Problem der algorithmischen Geometrie zur automatischen Platzierung von Labeln übersetzen

Geometrische Beschriftungsmodelle

Geg: n Punkte in der Ebene und für jeden Punkt ein Label repräsentiert als Rechteck (bounding box)

Ges: finde zulässige* Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



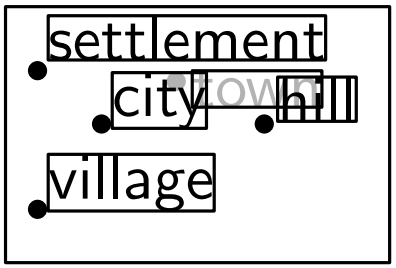
MaxNumber

Geometrische Beschriftungsmodelle

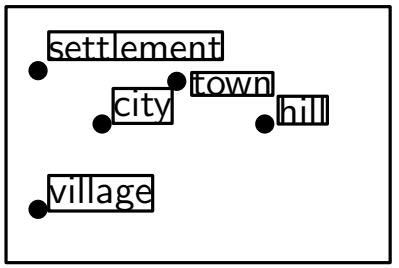
Geg: n Punkte in der Ebene und für jeden Punkt ein Label repräsentiert als Rechteck (bounding box)

Ges: finde zulässige* Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

oder finde zulässige* Beschriftung **aller** Label, so dass sich keine zwei Label schneiden und die **Schriftgröße** maximal ist (MaxSize)



MaxNumber



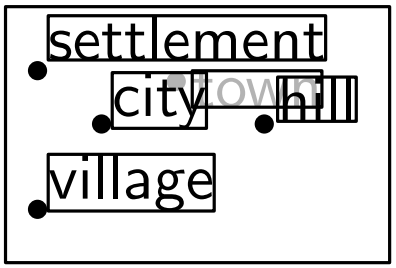
MaxSize

Geometrische Beschriftungsmodelle

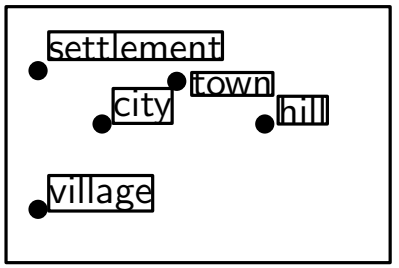
Geg: n Punkte in der Ebene und für jeden Punkt ein Label repräsentiert als Rechteck (bounding box)

Ges: finde zulässige* Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

oder finde zulässige* Beschriftung **aller** Label, so dass sich keine zwei Label schneiden und die **Schriftgröße** maximal ist (MaxSize)



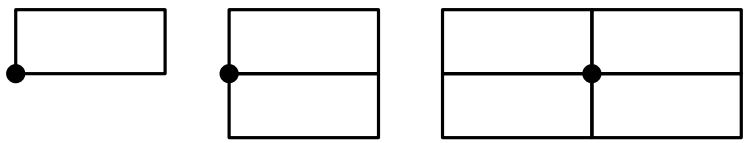
MaxNumber



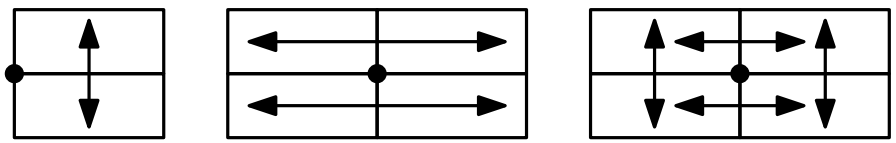
MaxSize

* Was ist eine **zulässige** Beschriftung?

diskrete Modelle



Slider-Modelle

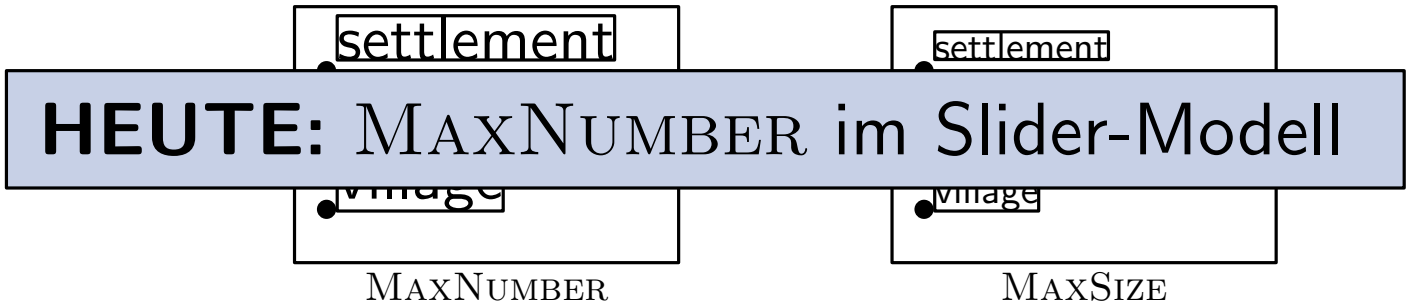


Geometrische Beschriftungsmodelle

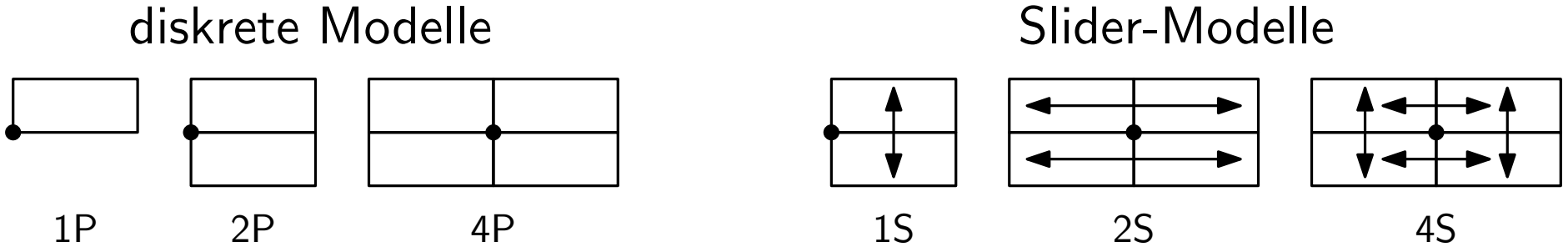
Geg: n Punkte in der Ebene und für jeden Punkt ein Label repräsentiert als Rechteck (bounding box)

Ges: finde zulässige* Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

oder finde zulässige* Beschriftung **aller** Label, so dass sich keine zwei Label schneiden und die **Schriftgröße** maximal ist (MaxSize)



* Was ist eine **zulässige** Beschriftung?



Teil 1: Komplexität

Komplexität

Ziel: Das 4-Slider MaxNumber Beschriftungsproblem ist NP-vollständig.

Idee: Reduktion von einer speziellen geometrischen 3-SAT Variante, sog. PLANARES 3-SAT

Planares 3-Sat

Erinnerung 3-Sat:
Gegeben eine Boolesche Formel $\varphi = c_1 \wedge \cdots \wedge c_m$ in konjunktiver Normalform mit drei Literalen pro Klausel $c_i = x \vee y \vee z$ ist es NP-vollständig zu entscheiden, ob φ eine erfüllende Belegung besitzt.

Planares 3-Sat

Erinnerung 3-Sat:
Gegeben eine Boolesche Formel $\varphi = c_1 \wedge \cdots \wedge c_m$ in konjunktiver Normalform mit drei Literalen pro Klausel $c_i = x \vee y \vee z$ ist es NP-vollständig zu entscheiden, ob φ eine erfüllende Belegung besitzt.

Def: Eine 3-Sat Formel φ heißt **planar**, falls der induzierte Variablen-Klausel-Graph $H_\varphi = (V, E)$ planar ist:

- V: Menge der Klauseln und Variablen
- E: Vorkommnisse von Variablen in Klauseln

Planares 3-Sat

Erinnerung 3-Sat:
Gegeben eine Boolesche Formel $\varphi = c_1 \wedge \cdots \wedge c_m$ in konjunktiver Normalform mit drei Literalen pro Klausel $c_i = x \vee y \vee z$ ist es NP-vollständig zu entscheiden, ob φ eine erfüllende Belegung besitzt.

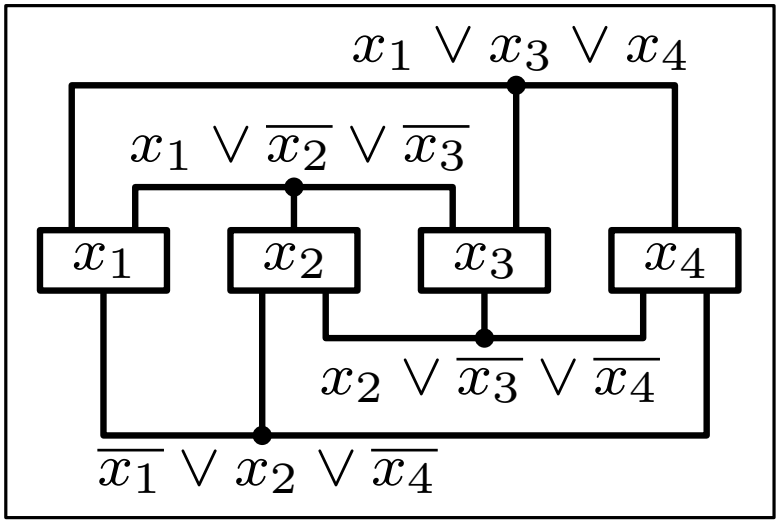
Def: Eine 3-Sat Formel φ heißt **planar**, falls der induzierte Variablen-Klausel-Graph $H_\varphi = (V, E)$ planar ist:

- V: Menge der Klauseln und Variablen
- E: Vorkommnisse von Variablen in Klauseln

Satz: Das 3-Sat Problem für planare Boolesche Formeln ist weiterhin NP-vollständig. (Lichtenstein '82)

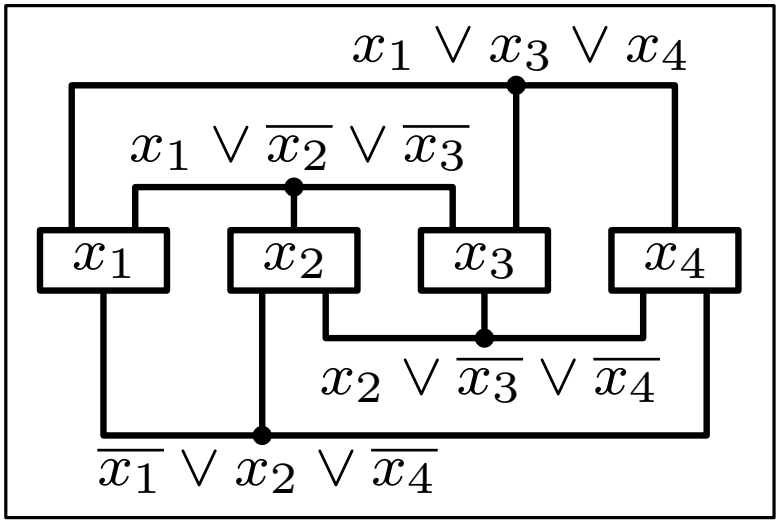
Zeichnung von H_φ

H_φ besitzt eine planare Zeichnung, bei der die Variablen auf einer Horizontalen angeordnet sind und die Klauseln E-förmig von oben oder unten zu den Variablen verbunden sind.



Zeichnung von H_φ

H_φ besitzt eine planare Zeichnung, bei der die Variablen auf einer Horizontalen angeordnet sind und die Klauseln E-förmig von oben oder unten zu den Variablen verbunden sind.



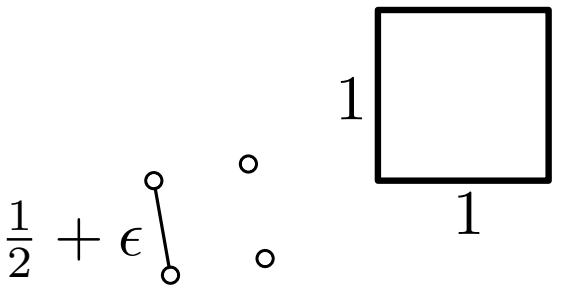
Reduktionsidee:

finde spezielle Punkt- und Labelmengen als **Gadgets** für Variablen und Klauseln in der Zeichnung, so dass gilt

φ erfüllbar $\Leftrightarrow$ alle Punkte in Labelinginstanz beschriftbar

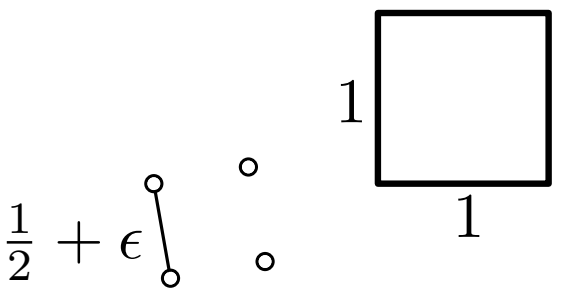
Variablengadget

- 4 Punkte als Quadratecken, leicht gedreht, Seitenlänge $1/2 + \epsilon$
- Label sind Quadrate der Länge 1



Variablengadget

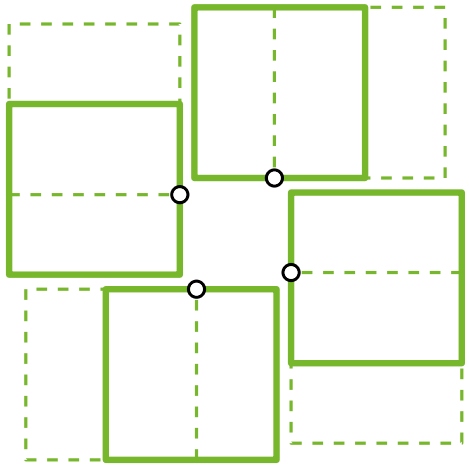
- 4 Punkte als Quadratecken, leicht gedreht, Seitenlänge $1/2 + \epsilon$
- Label sind Quadrate der Länge 1



Wie könnte man die vier Punkte beschriften?

Variablengadget

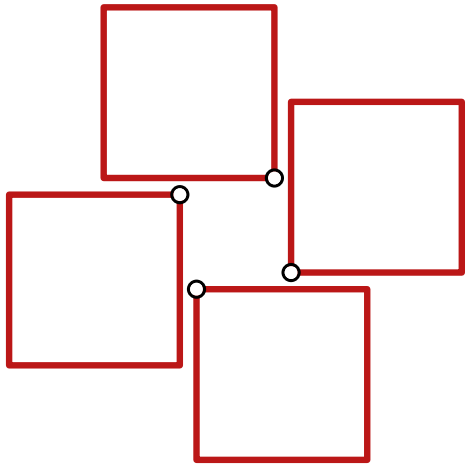
- 4 Punkte als Quadratecken, leicht gedreht, Seitenlänge $1/2 + \epsilon$
- Label sind Quadrate der Länge 1
- zwei fundamental verschiedene Beschriftungen möglich
- Kodierung der Wahrheitswerte



true (mit etwas Spiel)

Variablengadget

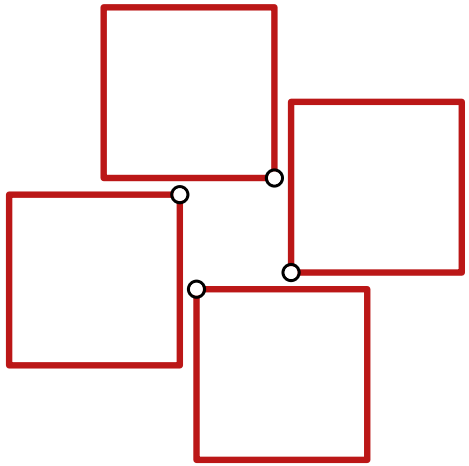
- 4 Punkte als Quadratecken, leicht gedreht, Seitenlänge $1/2 + \epsilon$
- Label sind Quadrate der Länge 1
- zwei fundamental verschiedene Beschriftungen möglich
- Kodierung der Wahrheitswerte



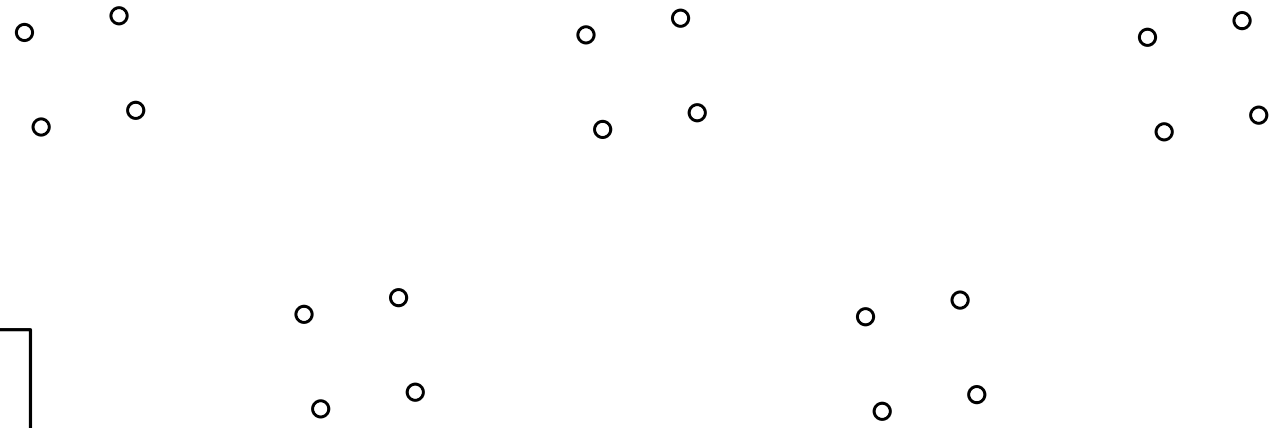
false (fast fest)

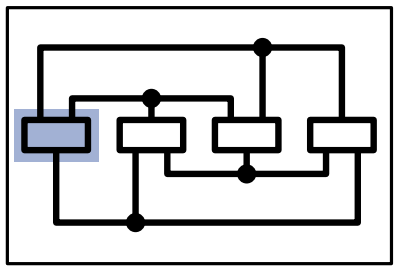
Variablengadget

- 4 Punkte als Quadratecken, leicht gedreht, Seitenlänge $1/2 + \epsilon$
- Label sind Quadrate der Länge 1
- zwei fundamental verschiedene Beschriftungen möglich
- Kodierung der Wahrheitswerte
- Variablen-Gadgets im zick-zack Muster



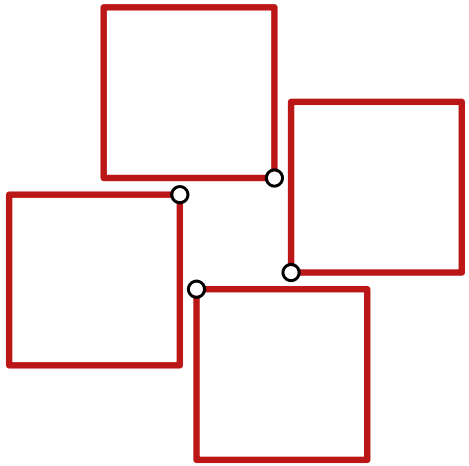
false (fast fest)



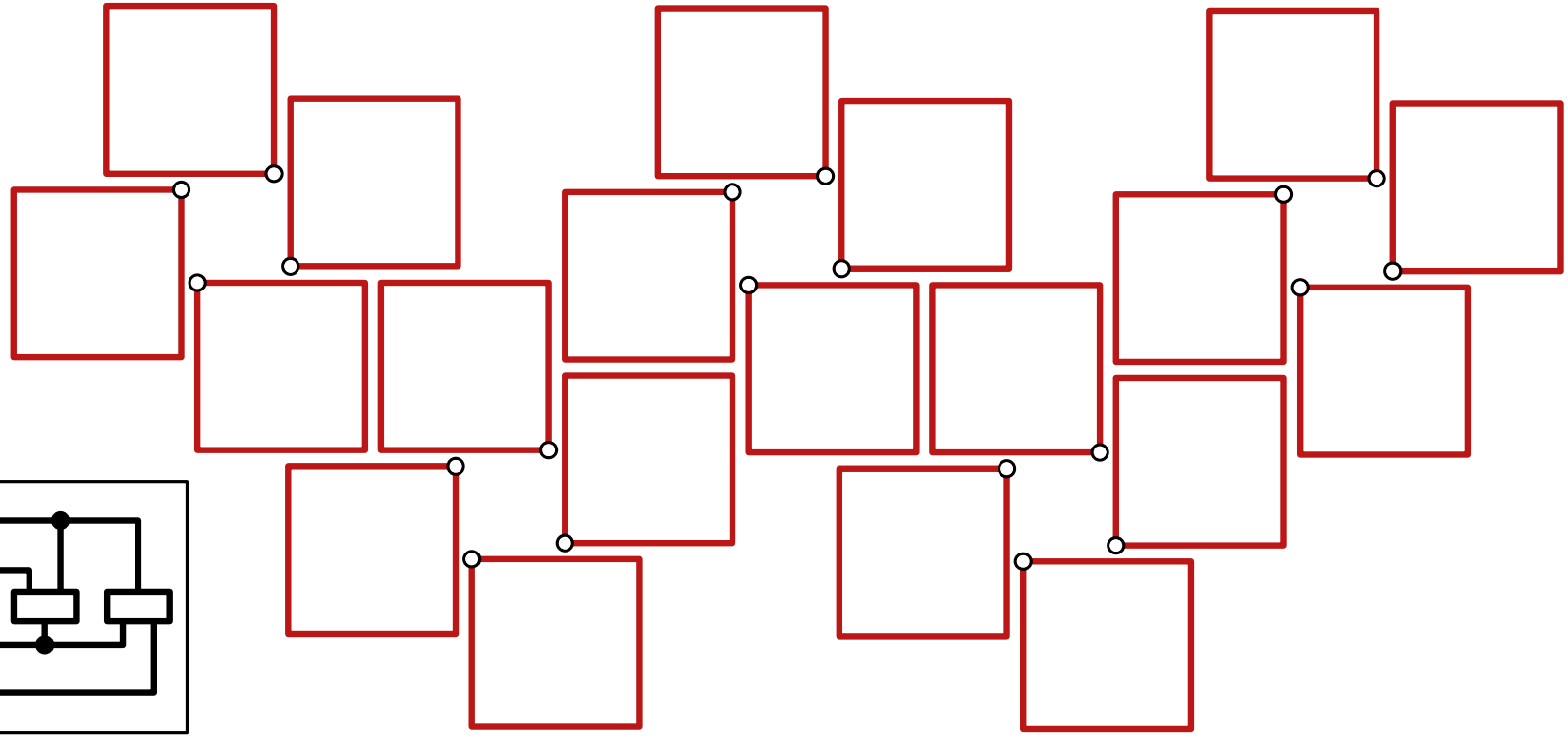


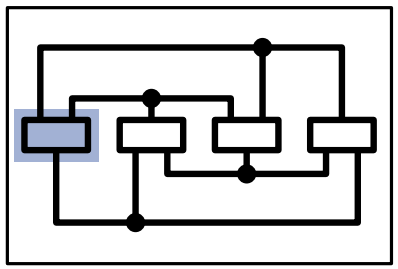
Variablengadget

- 4 Punkte als Quadratecken, leicht gedreht, Seitenlänge $1/2 + \epsilon$
- Label sind Quadrate der Länge 1
- zwei fundamental verschiedene Beschriftungen möglich
- Kodierung der Wahrheitswerte
- Variablen-Gadgets im zick-zack Muster



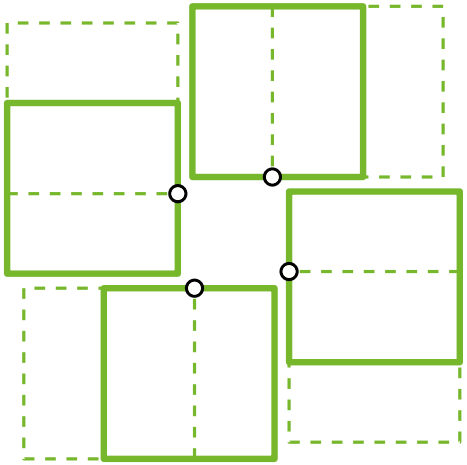
false (fast fest)



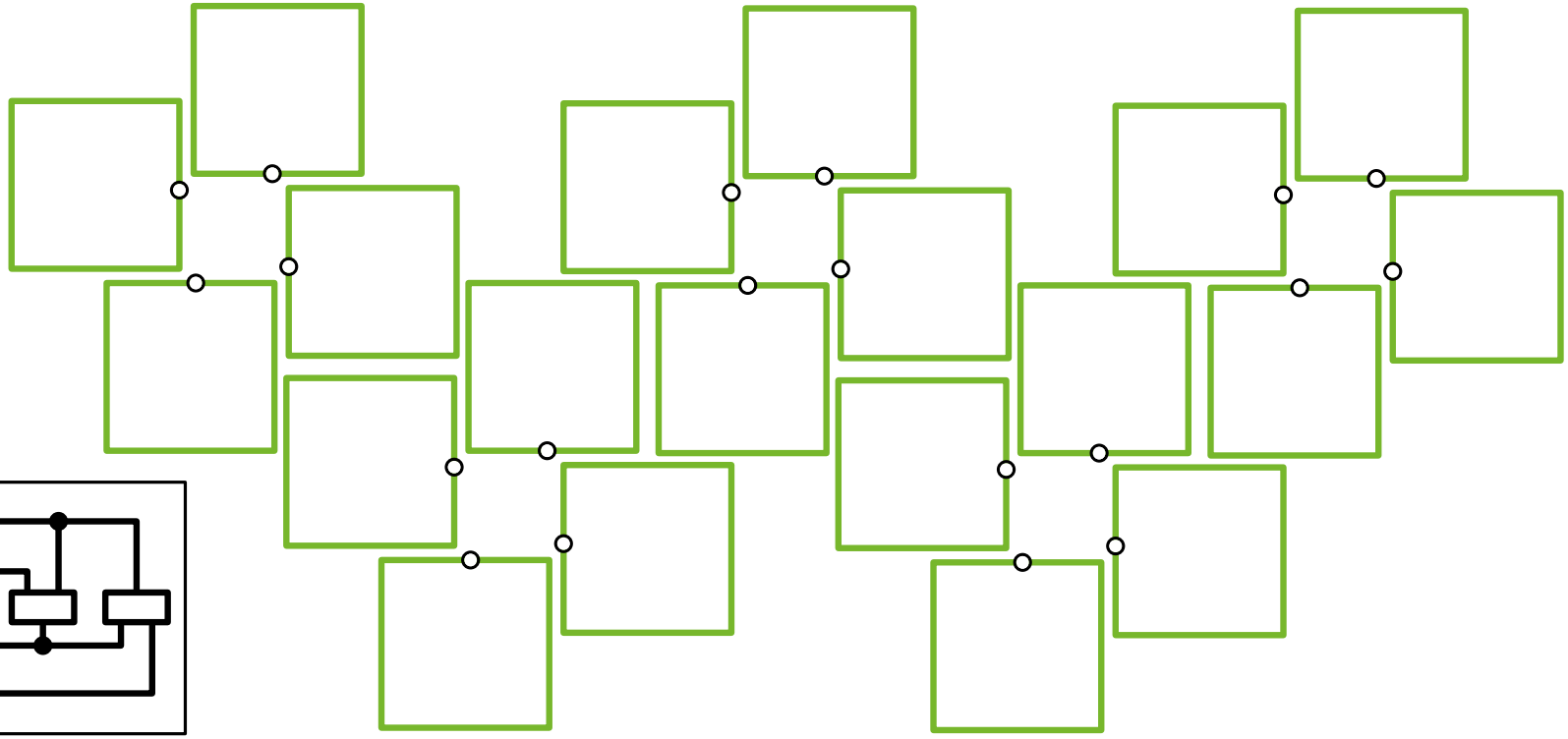


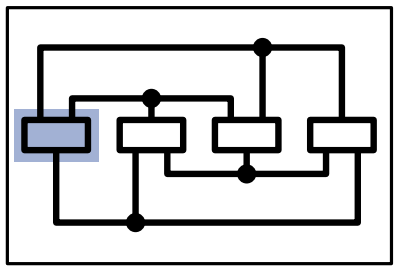
Variablengadget

- 4 Punkte als Quadratecken, leicht gedreht, Seitenlänge $1/2 + \epsilon$
- Label sind Quadrate der Länge 1
- zwei fundamental verschiedene Beschriftungen möglich
- Kodierung der Wahrheitswerte
- Variablen-Gadgets im zick-zack Muster



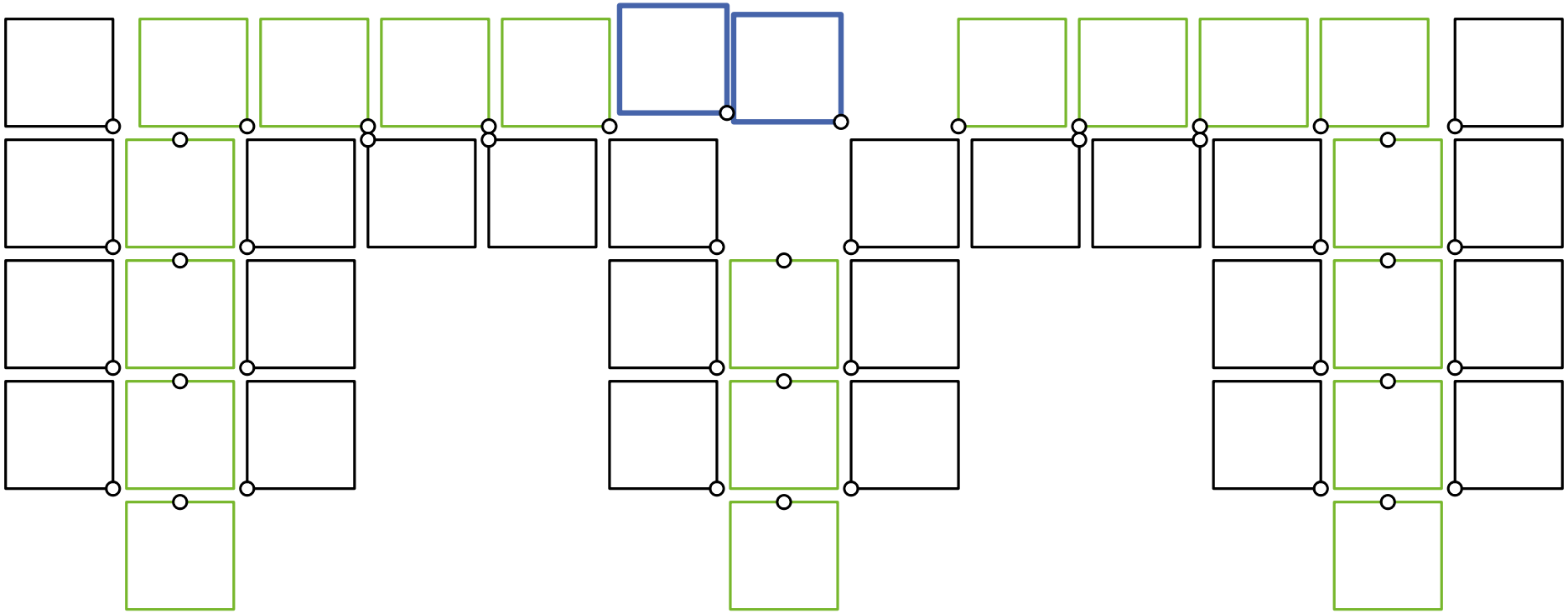
true (mit etwas Spiel)





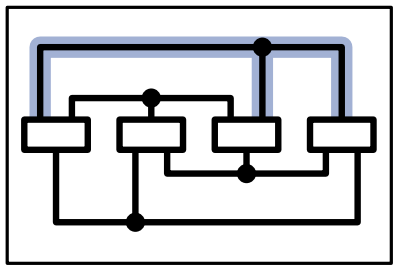
Klauselgadget

Nachbildung der E-Form der Klauseln aus Zeichnung von H_φ



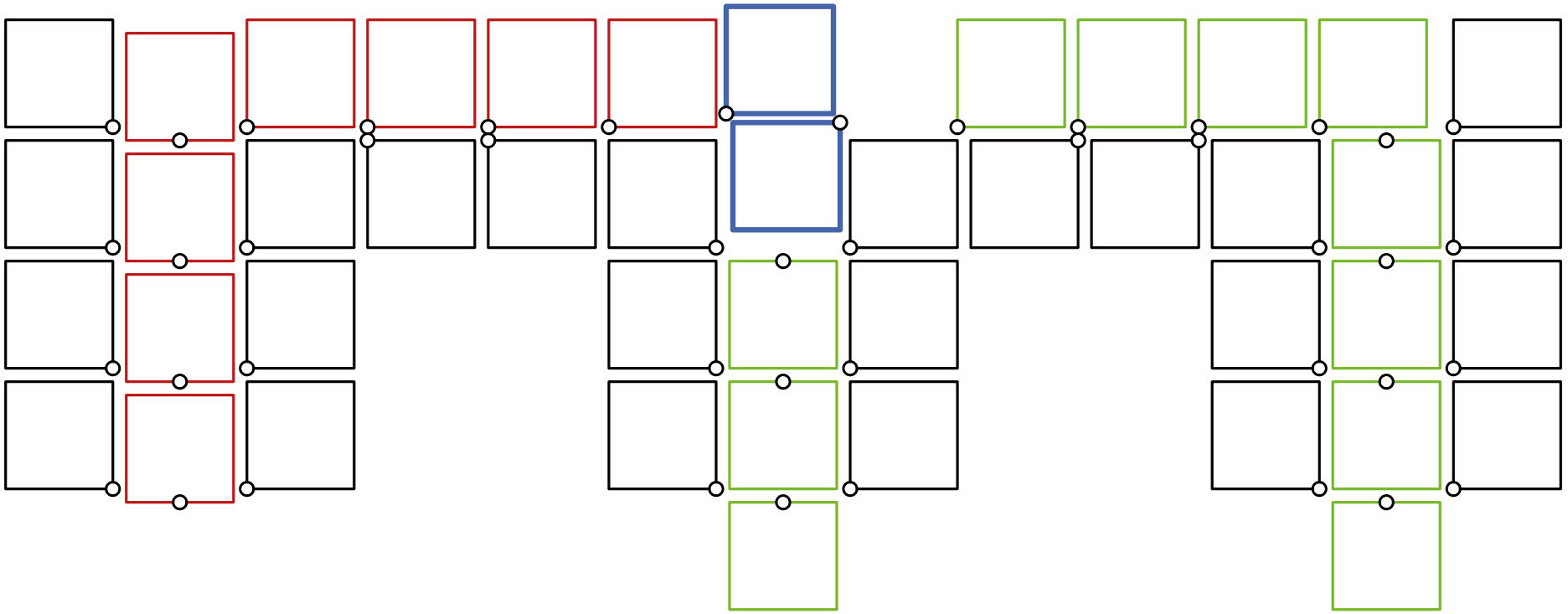
Funktionsweise:

- falsche Literale „drücken“ Labelkette nach oben
- wahre Literale ohne Druck
- zwei Label im Klauselzentrum selektieren wahres Literal
- Selektoren schneiden sich $\Leftrightarrow$ alle Literale falsch



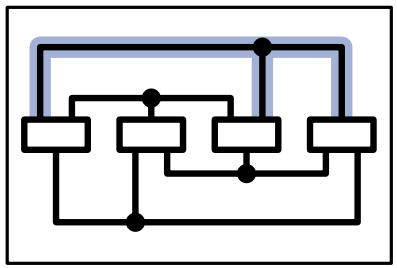
Klauselgadget

Nachbildung der E-Form der Klauseln aus Zeichnung von H_φ



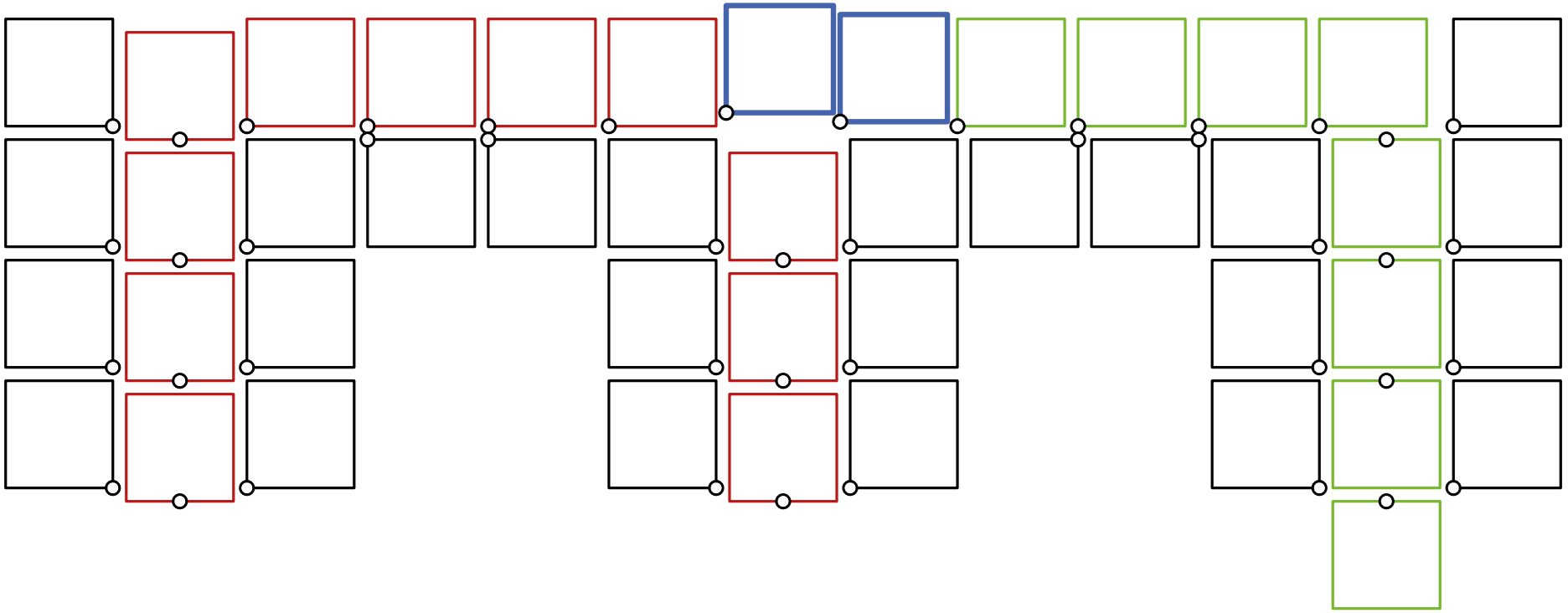
Funktionsweise:

- falsche Literale „drücken“ Labelkette nach oben
- wahre Literale ohne Druck
- zwei Label im Klauselzentrum selektieren wahres Literal
- Selektoren schneiden sich $\Leftrightarrow$ alle Literale falsch



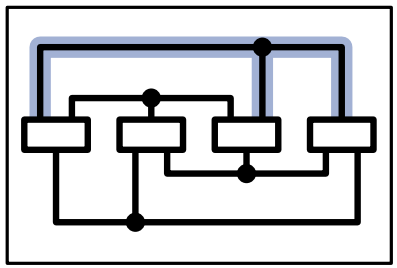
Klauselgadget

Nachbildung der E-Form der Klauseln aus Zeichnung von H_φ



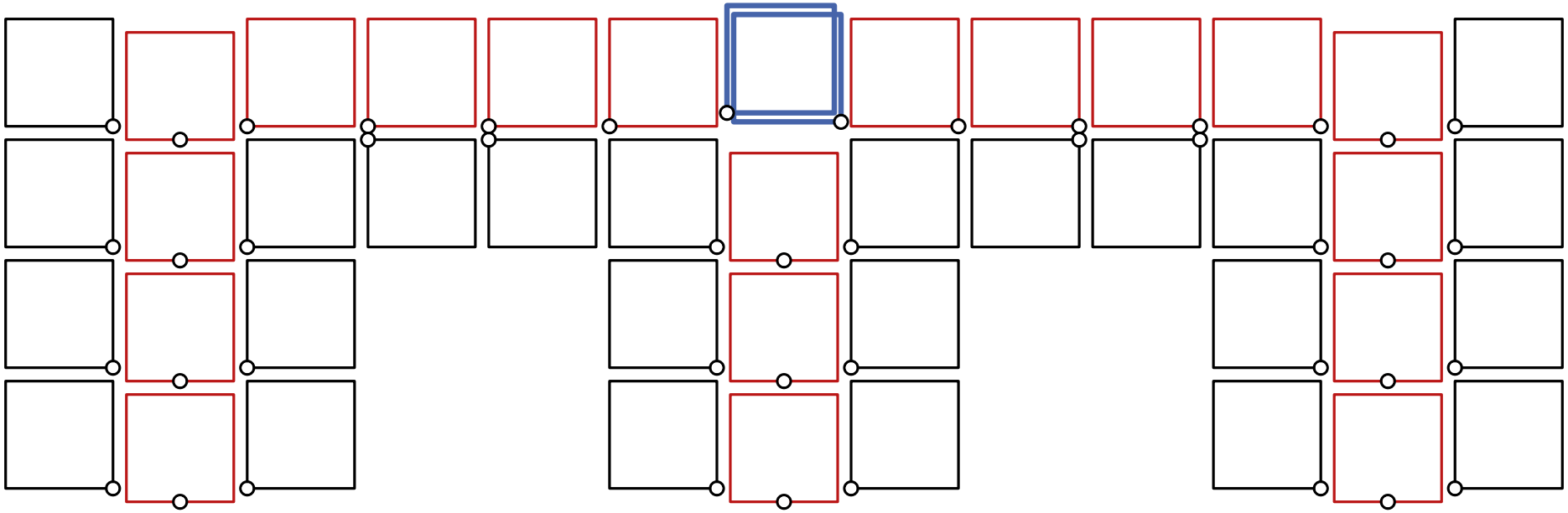
Funktionsweise:

- falsche Literale „drücken“ Labelkette nach oben
- wahre Literale ohne Druck
- zwei Label im Klauselzentrum selektieren wahres Literal
- Selektoren schneiden sich $\Leftrightarrow$ alle Literale falsch



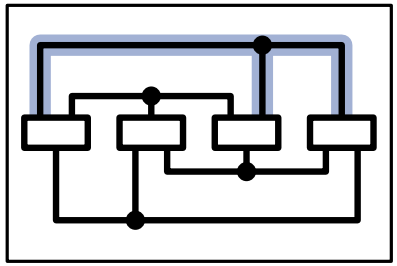
Klauselgadget

Nachbildung der E-Form der Klauseln aus Zeichnung von H_φ



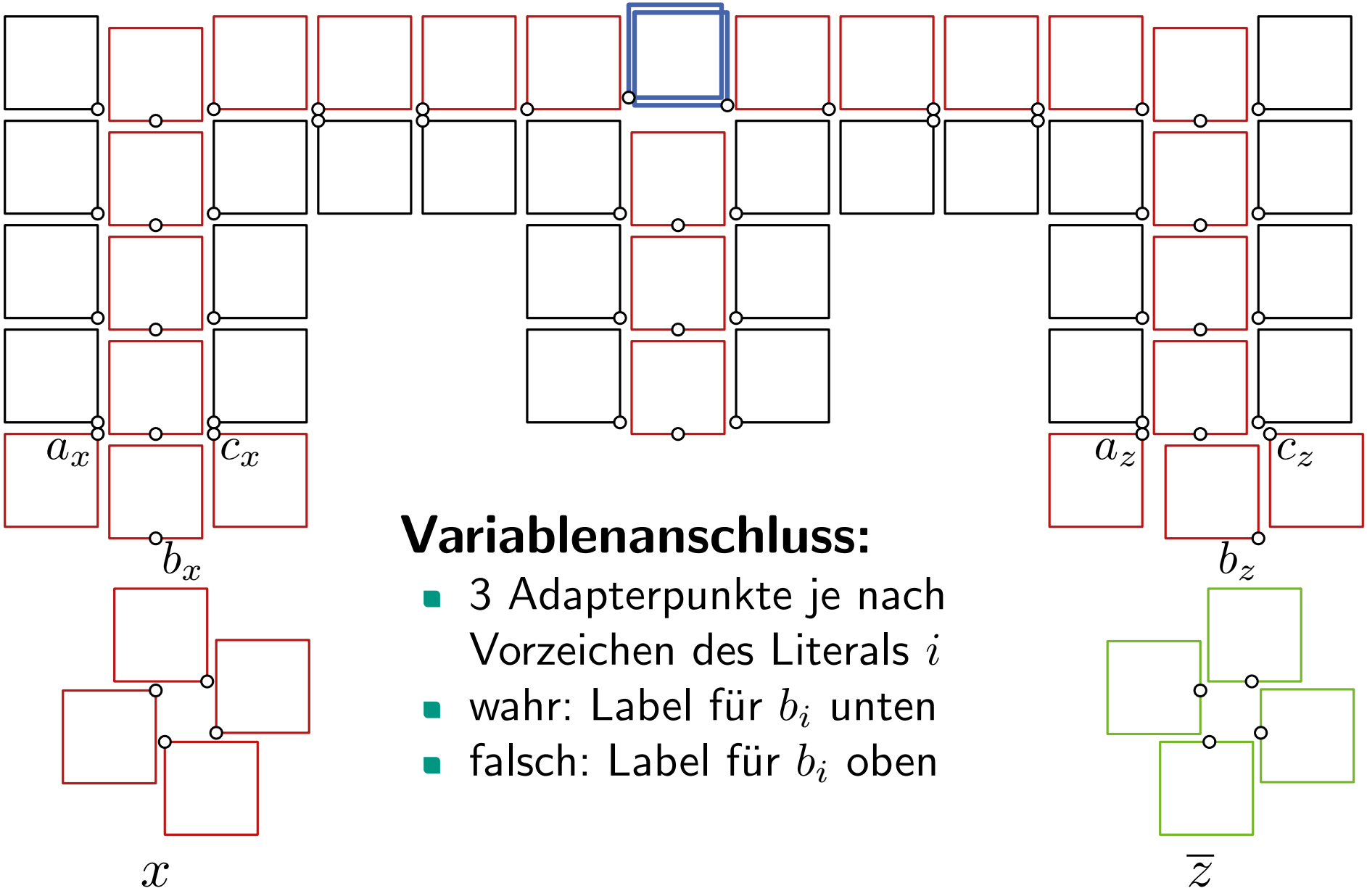
Funktionsweise:

- falsche Literale „drücken“ Labelkette nach oben
- wahre Literale ohne Druck
- zwei Label im Klauselzentrum selektieren wahres Literal
- Selektoren schneiden sich $\Leftrightarrow$ alle Literale falsch



Klauselgadget

Nachbildung der E-Form der Klauseln aus Zeichnung von H_φ

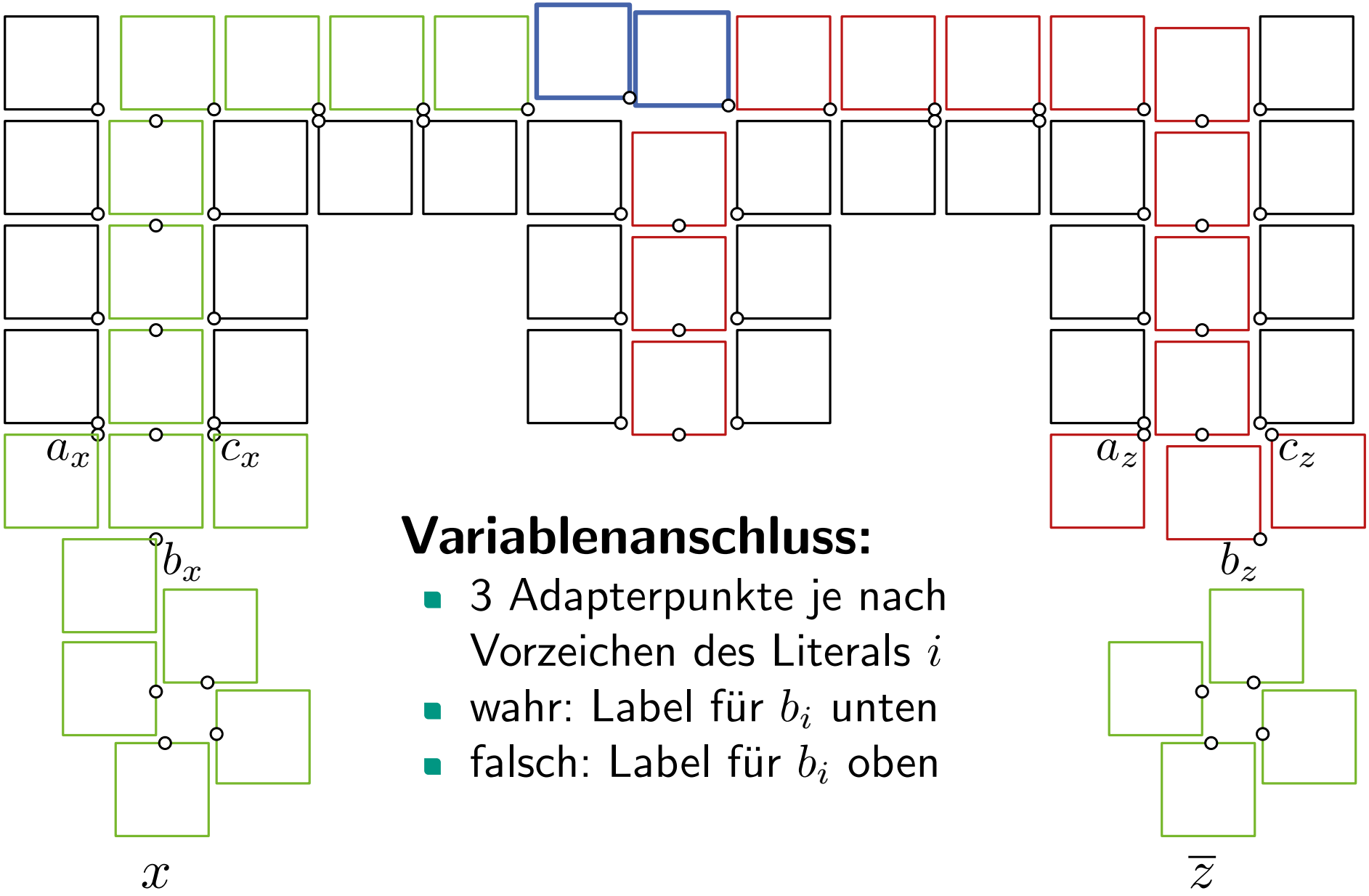


Variablenanschluss:

- 3 Adapterpunkte je nach Vorzeichen des Literals i
- wahr: Label für b_i unten
- falsch: Label für b_i oben

Klauselgadget

Nachbildung der E-Form der Klauseln aus Zeichnung von H_φ

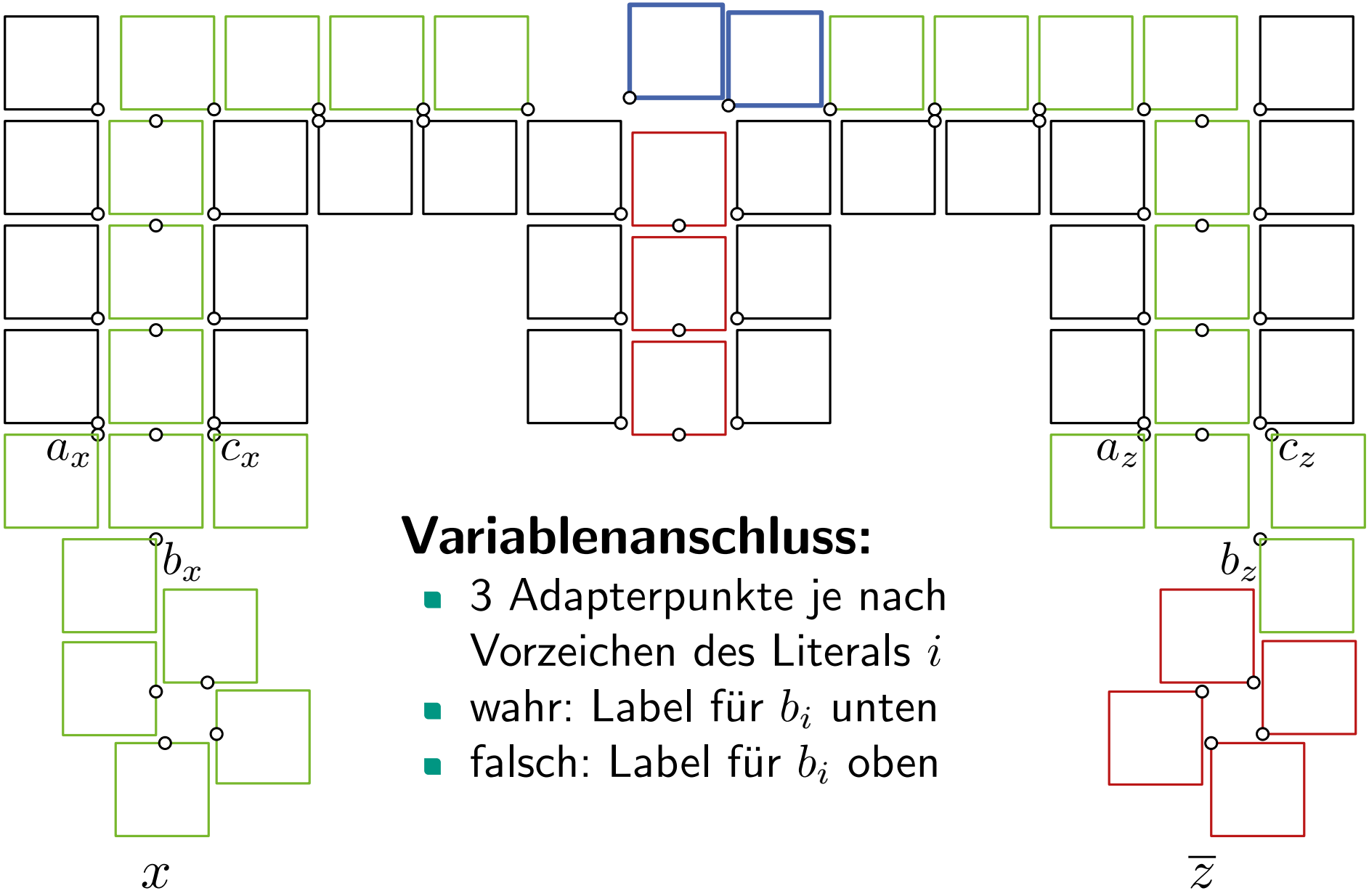


Variablenanschluss:

- 3 Adapterpunkte je nach Vorzeichen des Literals i
- wahr: Label für b_i unten
- falsch: Label für b_i oben

Klauselgadget

Nachbildung der E-Form der Klauseln aus Zeichnung von H_φ

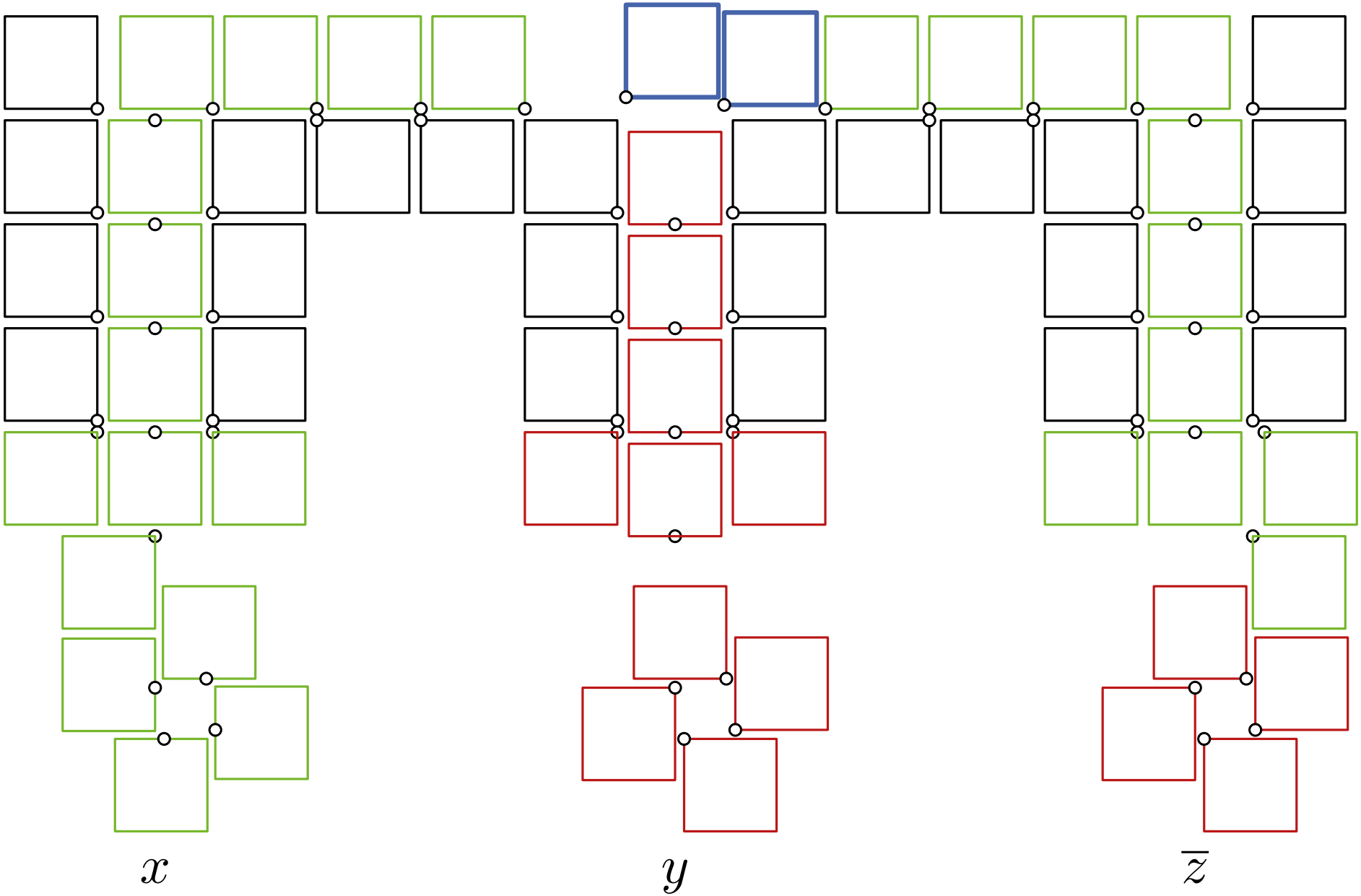


Variablenanschluss:

- 3 Adapterpunkte je nach Vorzeichen des Literals i
- wahr: Label für b_i unten
- falsch: Label für b_i oben

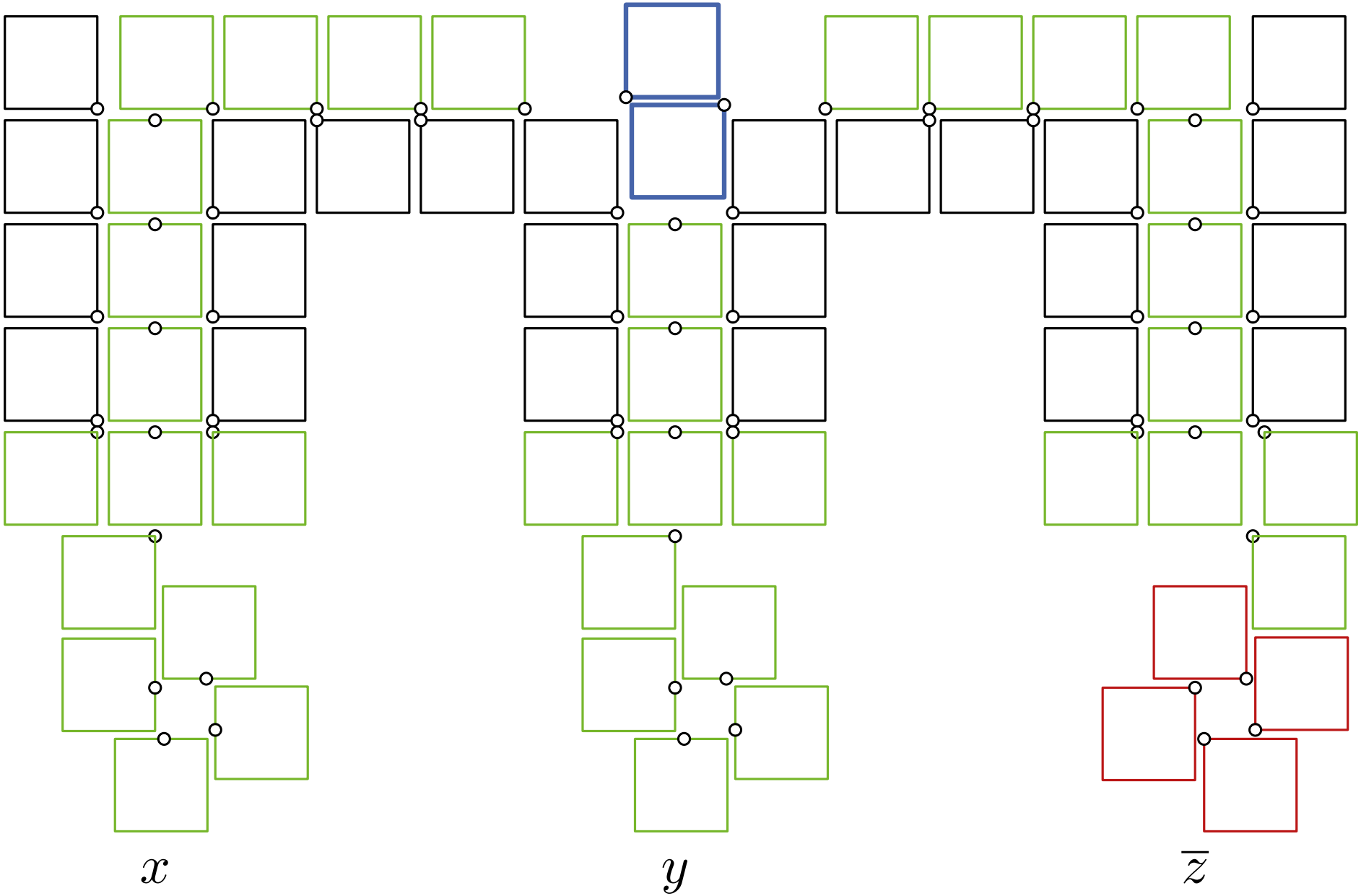
Klauselgadget

Nachbildung der E-Form der Klauseln aus Zeichnung von H_φ



Klauselgadget

Nachbildung der E-Form der Klauseln aus Zeichnung von H_φ



Komplexität

Satz 1: Es ist NP-vollständig zu entscheiden, ob in einer gegebenen Instanz I im 4-Slider Modell alle Punkte beschriftet werden können, selbst wenn alle Label Einheitsquadrate sind. (van Kreveld, Strijk, Wolff '99)

Komplexität

Satz 1: Es ist NP-vollständig zu entscheiden, ob in einer gegebenen Instanz I im 4-Slider Modell alle Punkte beschriftet werden können, selbst wenn alle Label Einheitsquadrate sind. (van Kreveld, Strijk, Wolff '99)

Beweis:

- nach Konstruktion gilt
  Formel φ erfüllbar $\Leftrightarrow$ Instanz I_φ beschriftbar
- Größe von I_φ quadratisch in Klauselanzahl

Komplexität

Satz 1: Es ist NP-vollständig zu entscheiden, ob in einer gegebenen Instanz I im 4-Slider Modell alle Punkte beschriftet werden können, selbst wenn alle Label Einheitsquadrate sind. (van Kreveld, Strijk, Wolff '99)

Beweis:

- nach Konstruktion gilt
  Formel φ erfüllbar $\Leftrightarrow$ Instanz I_φ beschriftbar
- Größe von I_φ quadratisch in Klauselanzahl

Warum liegt das Problem in $\mathcal{NP}$?

Teil 2: Approximationsalgorithmen

Greedy-Algorithmus

Geg: Punktmenge $P = \{p_1, \ldots, p_n\}$, Menge rechteckiger Label $L = \{l_1, \ldots l_n\}$ mit Höhe 1 und variabler Breite

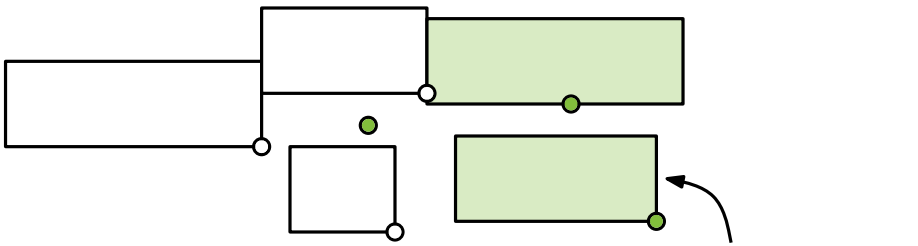
Modell: 4-Slider

Greedy-Algorithmus

Geg: Punktmenge $P = \{p_1, \ldots, p_n\}$, Menge rechteckiger Label $L = \{l_1, \ldots l_n\}$ mit Höhe 1 und variabler Breite

Modell: 4-Slider

Def: Für eine Menge P' von Punkten mit bereits platzierten Labeln heißt ein Label l für einen Punkt $p \in P \setminus P'$ **linkestes** Label, falls seine rechte Kante unter allen noch konfliktfrei platzierbaren Labeln am weitesten links liegt.



Greedy-Algorithmus

Geg: Punktmenge $P = \{p_1, \ldots, p_n\}$, Menge rechteckiger Label $L = \{l_1, \ldots l_n\}$ mit Höhe 1 und variabler Breite

Modell: 4-Slider

Def: Für eine Menge P' von Punkten mit bereits platzierten Labeln heißt ein Label l für einen Punkt $p \in P \setminus P'$ **linkestes** Label, falls seine rechte Kante unter allen noch konfliktfrei platzierbaren Labeln am weitesten links liegt.

Algorithmus Greedy4S(P,L)

while linkestes Label l existiert **do**
  platziere l linkest möglich

Lemma 1: Greedy4S berechnet eine Faktor- Approximation.

Versuchen Sie den Approximationsfaktor zu bestimmen!

Greedy-Algorithmus

Geg: Punktmenge $P = \{p_1, \ldots, p_n\}$, Menge rechteckiger Label $L = \{l_1, \ldots l_n\}$ mit Höhe 1 und variabler Breite

Modell: 4-Slider

Def: Für eine Menge P' von Punkten mit bereits platzierten Labeln heißt ein Label l für einen Punkt $p \in P \setminus P'$ **linkestes** Label, falls seine rechte Kante unter allen noch konfliktfrei platzierbaren Labeln am weitesten links liegt.

Algorithmus Greedy4S(P,L)

while linkestes Label l existiert **do**
  platziere l linkest möglich

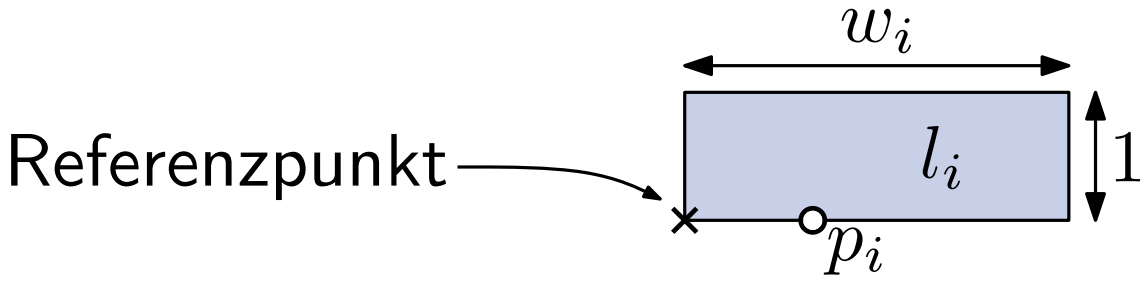
Lemma 1: Greedy4S berechnet eine Faktor-1/2 Approximation.

Laufzeit

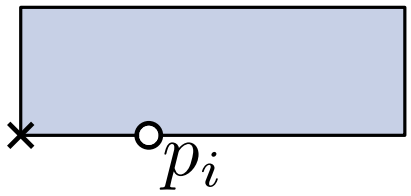
Eine naive Implementierung von Greedy4S benötigt $O(n^3)$ Zeit.

Ziel: Nutze geeignete geometrische Datenstrukturen um die Laufzeit auf $O(n \log n)$ zu senken.

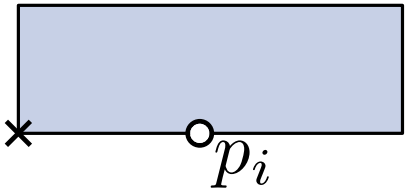
Notation



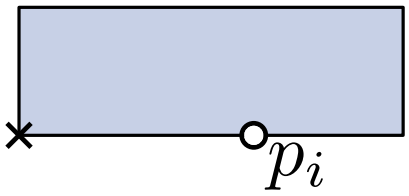
Notation



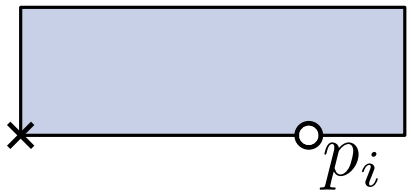
Notation



Notation



Notation

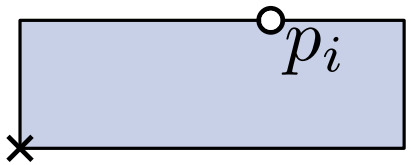


Notation

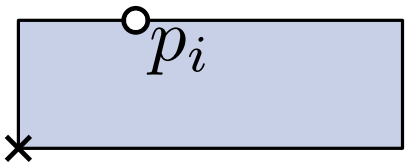
Notation

Notation

Notation

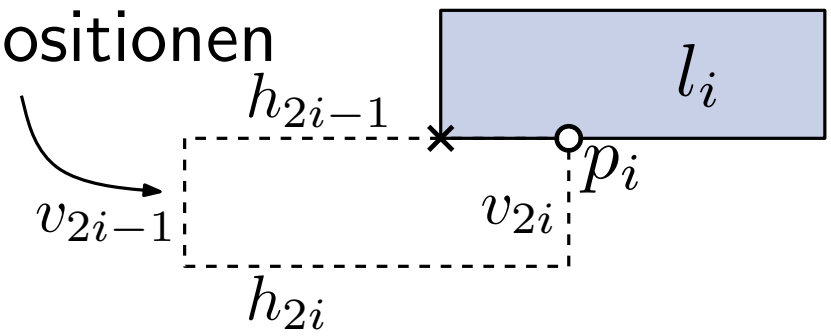


Notation



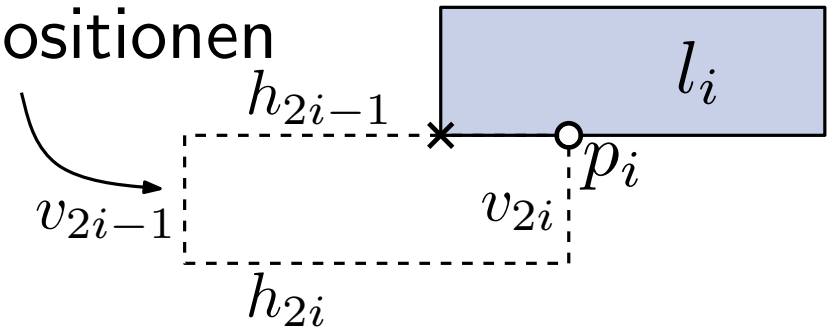
Notation

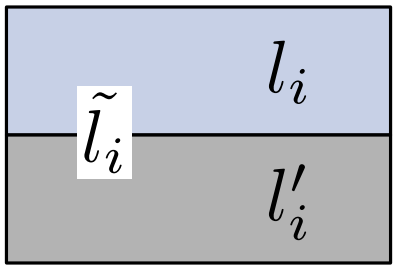
mögl. Positionen



Notation

mögl. Positionen

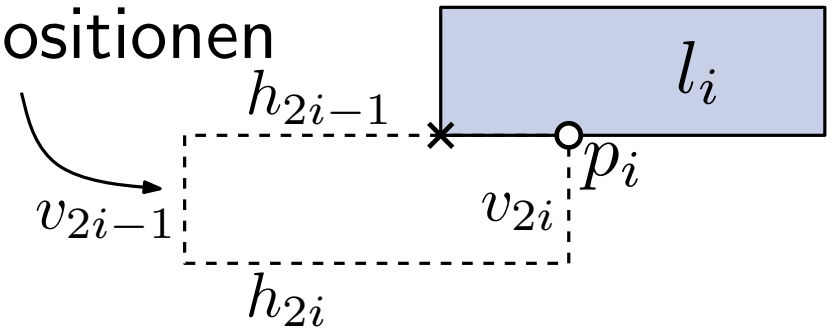


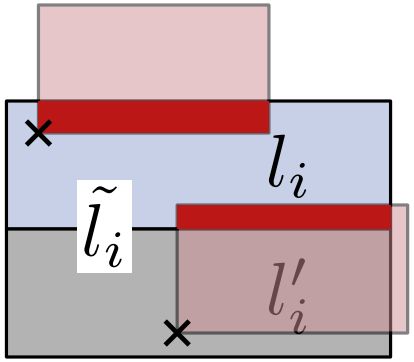


Label l_i und **erweitertes Label** $\tilde{l}_i$

Notation

mögl. Positionen



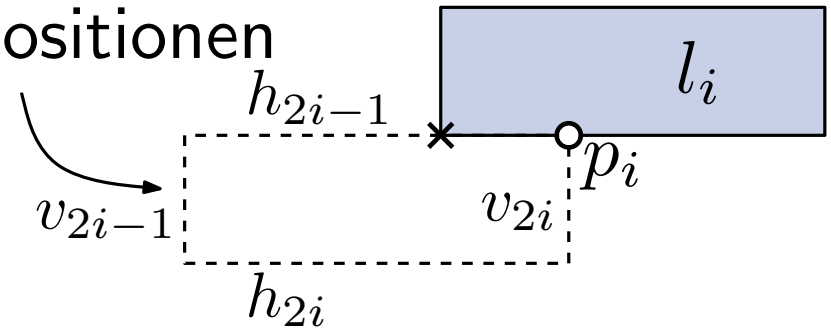


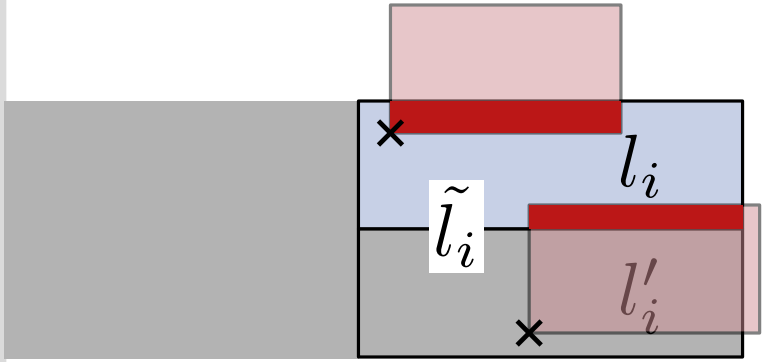
Label l_i und **erweitertes Label** $\tilde{l}_i$

kein Referenzpunkt darf in $\tilde{l}_i$ liegen

Notation

mögl. Positionen



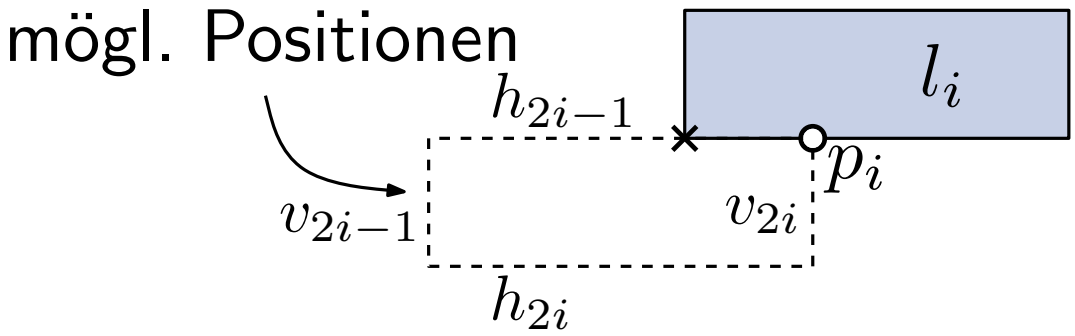


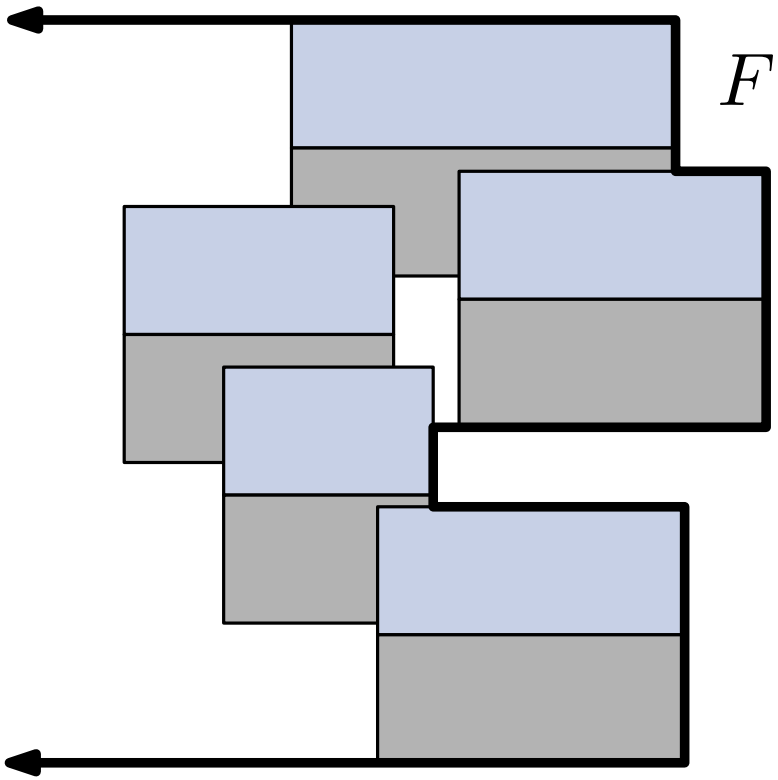
Label l_i und **erweitertes Label** $\tilde{l}_i$

kein Referenzpunkt darf in $\tilde{l}_i$ liegen

da immer linkestes Label platziert wird, kein Referenzpunkt links von $\tilde{l}_i$

Notation

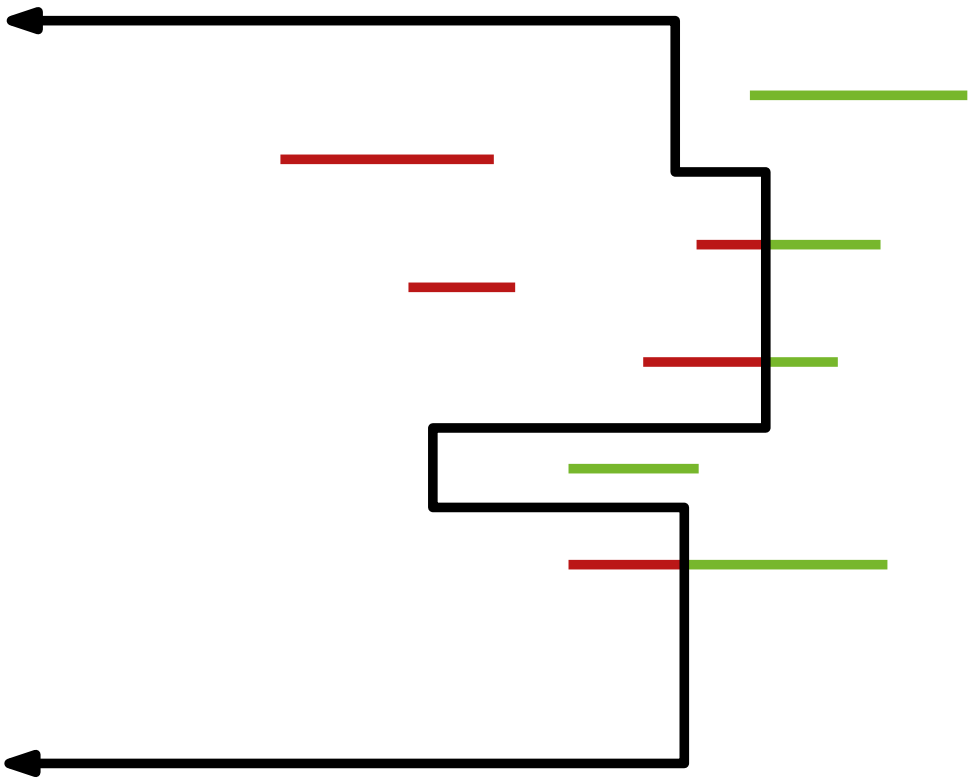




Grenze F als *right envelope* der platzierten Label

Grenze und linkestes Label

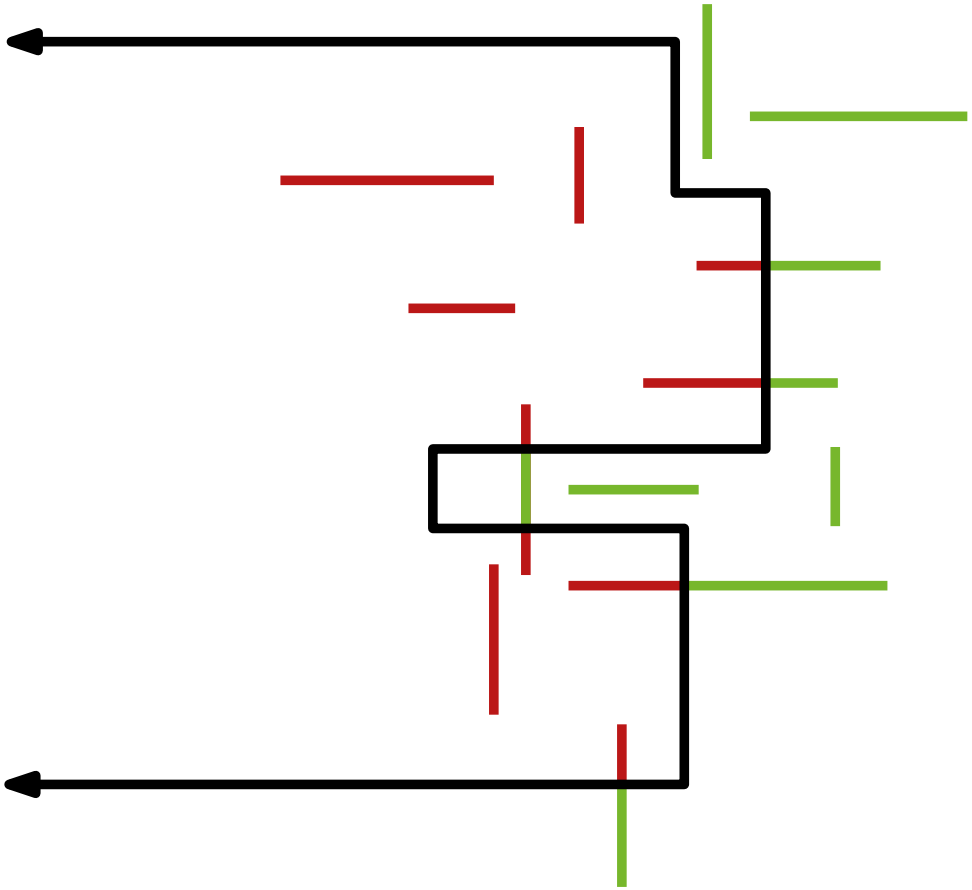
Zur Bestimmung des nächsten linkesten Labels genügt es F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.



- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken
- $H_{\text{right}} \subseteq H$: rechts von F
- $H_{\text{int}} \subseteq H$: schneiden F

Grenze und linkestes Label

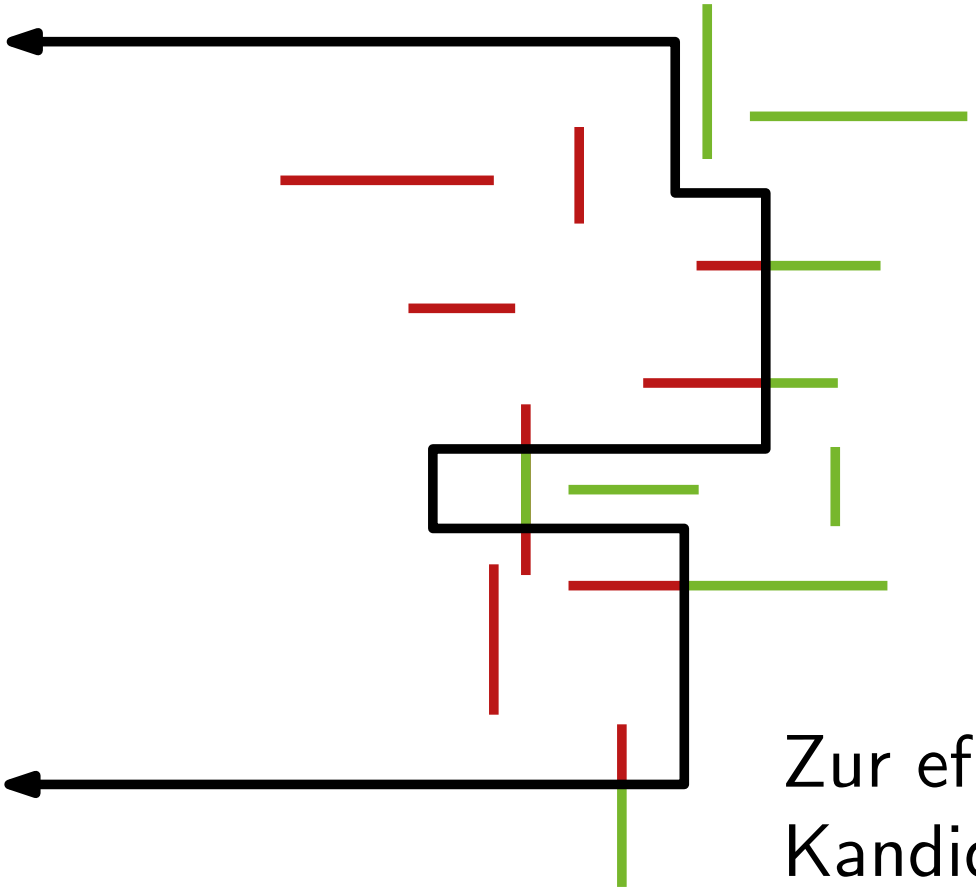
Zur Bestimmung des nächsten linkesten Labels genügt es F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.



- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken
- $H_{\text{right}} \subseteq H$: rechts von F
- $H_{\text{int}} \subseteq H$: schneiden F
- $V_{\text{int,right}} \subseteq V$: enthalten Punkt rechts von F
- linkeste Punkte im grünen Bereich der Strecken sind zulässige Kandidaten

Grenze und linkestes Label

Zur Bestimmung des nächsten linkesten Labels genügt es F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.



- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken
- $H_{\text{right}} \subseteq H$: rechts von F
- $H_{\text{int}} \subseteq H$: schneiden F
- $V_{\text{int,right}} \subseteq V$: enthalten Punkt rechts von F
- linkeste Punkte im grünen Bereich der Strecken sind zulässige Kandidaten

Zur effizienten Verwaltung der linkesten Kandidaten benötigen wir mehrere geometrische Datenstrukturen.

Datenstrukturen

1) Red-Black Trees

- selbstbalancierender binärer Suchbaum der Größe $O(n)$
- Suchen, Einfügen, Löschen, Auftrennen, Konkatenieren in $O(\log n)$ Zeit

Datenstrukturen

1) Red-Black Trees

- selbstbalancierender binärer Suchbaum der Größe $O(n)$
- Suchen, Einfügen, Löschen, Auftrennen, Konkatenieren in $O(\log n)$ Zeit

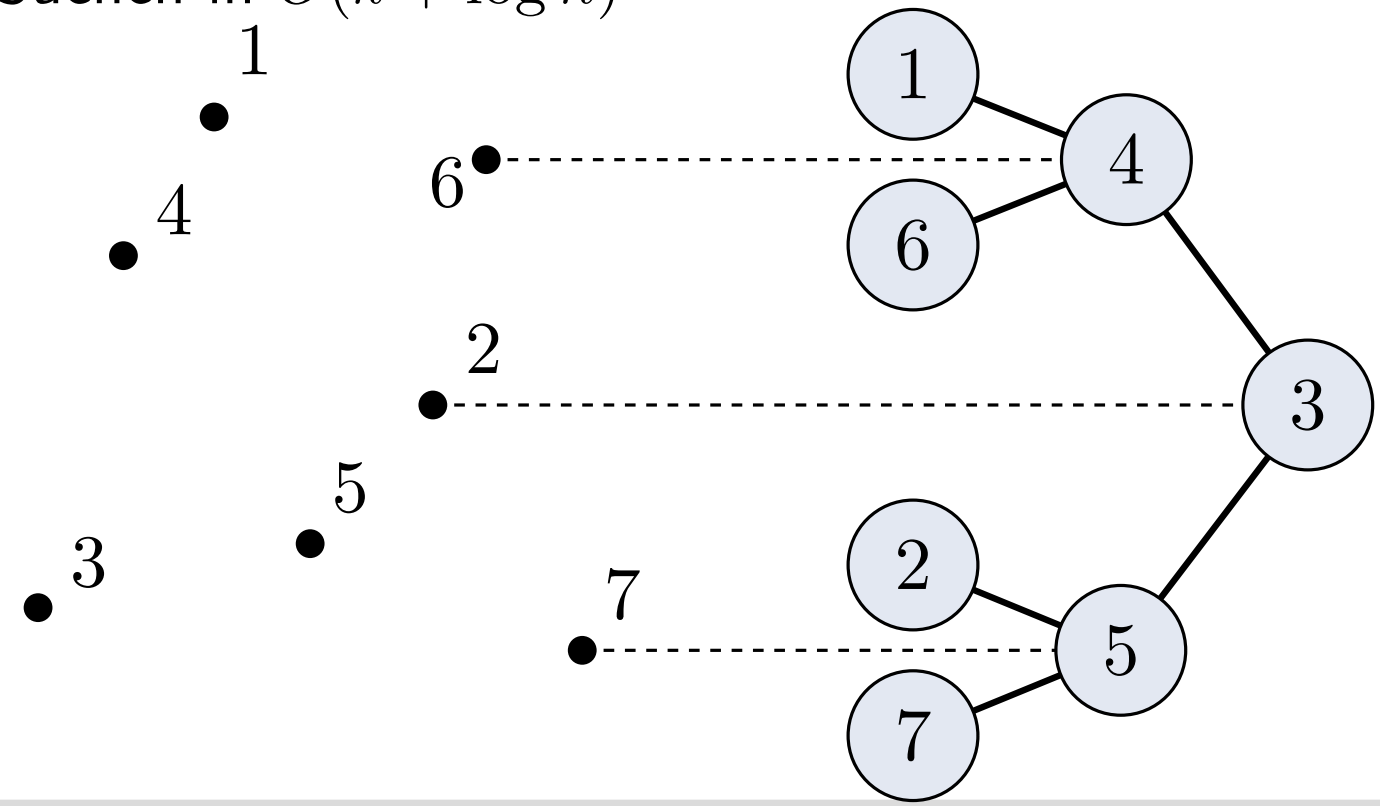
2) Heaps

- baumbasierte Datenstruktur mit heap-Eigenschaft
- key(parent(A)) $\leq$ key(A) für alle A
- Größe $O(n)$
- find-min in $O(1)$
- delete-min, Einfügen, Löschen in $O(\log n)$

Karlsruhe Institute of Technology

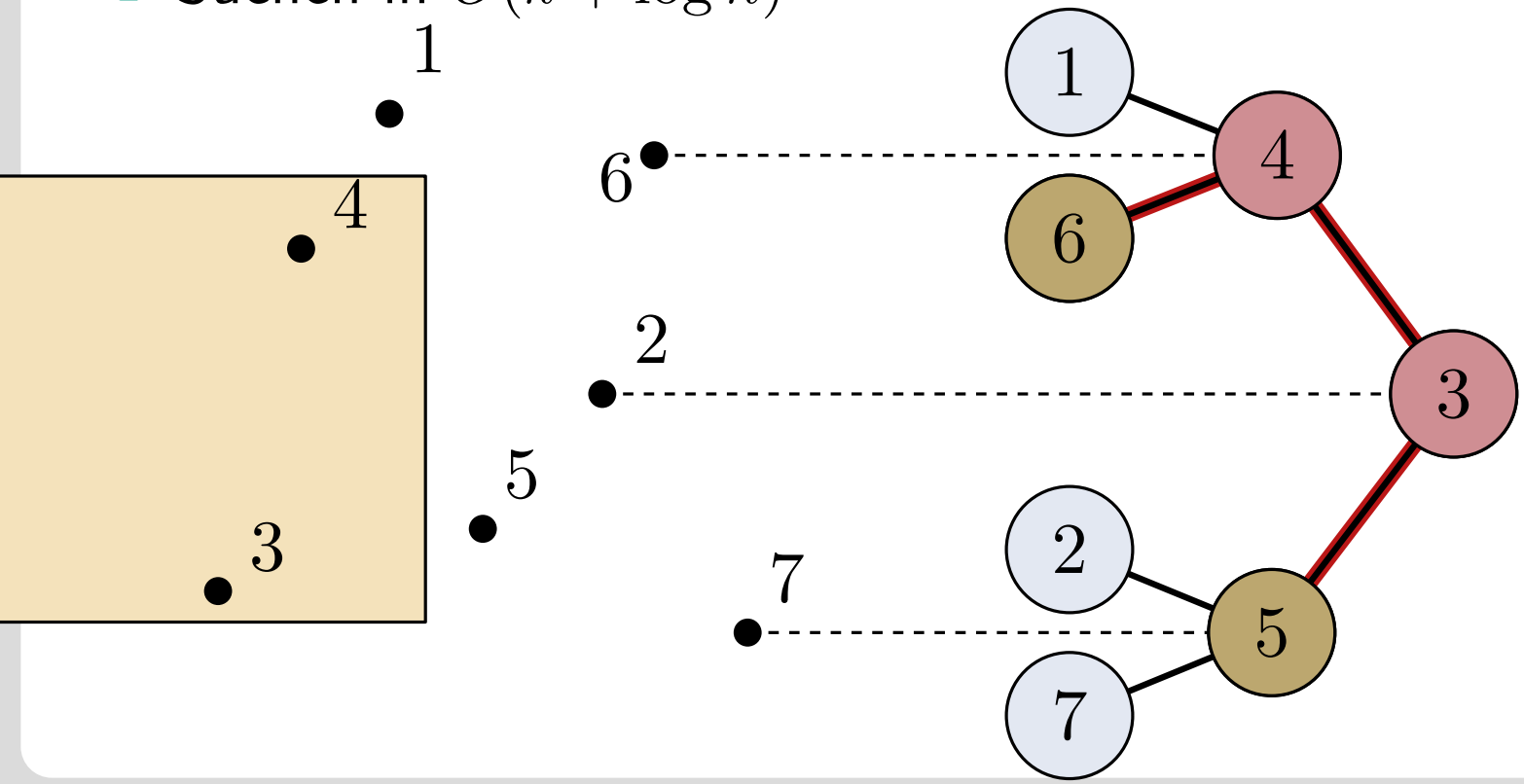
3) Priority Search Trees

- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y
- Größe $O(n)$
- Einfügen, Löschen in $O(\log n)$
- Suchen in $O(k + \log n)$



3) Priority Search Trees

- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y
- Größe $O(n)$
- Einfügen, Löschen in $O(\log n)$
- Suchen in $O(k + \log n)$



Horizontale Strecken

1) Menge H_{right}

- speichere für jede Strecke in H_{right} deren rechten Endpunkt (= linkest mögliche Position der rechten Labelseite)
- Datenstruktur: min-Heap $\mathcal{H}_{\text{right}}$

Horizontale Strecken

1) Menge H_{right}

- speichere für jede Strecke in H_{right} deren rechten Endpunkt ($=$ linkest mögliche Position der rechten Labelseite)
- Datenstruktur: min-Heap $\mathcal{H}_{\text{right}}$

2) Menge H_{int}

- Red-Black Tree $\mathcal{T}_i$ für jedes vertikale Segment f_i von F
- speichere Strecken aus H_{int}, die f_i schneiden sortiert nach y-Koordinaten in den Blättern von $\mathcal{T}_i$
- speichere zusätzlich Labelbreite an jedem Blatt
- an inneren Knoten minimale Labelbreite im Teilbaum
- min-Heap $\mathcal{H}_{\text{int}}$ für linkestes Label in jedem $\mathcal{T}_i$

Vertikale Strecken

3) Menge $V_{\text{int,right}}$

- speichere Obermenge V' mit $V_{\text{int,right}} \subseteq V'$
- min-Heap $\mathcal{H}_V$ der Menge V' geordnet nach x-Koordinaten
- falls Minimum in $\mathcal{H}_V$ nicht in $V_{\text{int,right}}$ liegt, verwerfe es und wähle nächstes Minimum

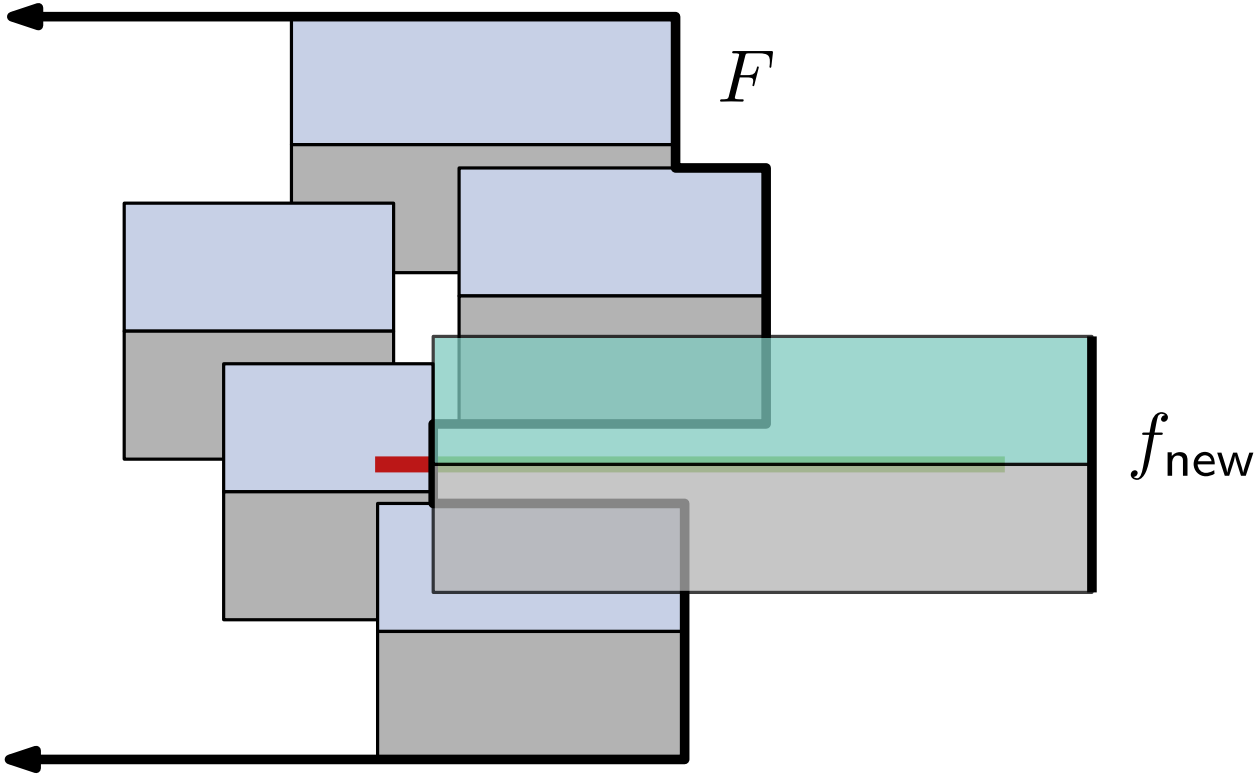
Vertikale Strecken

3) Menge $V_{\text{int,right}}$

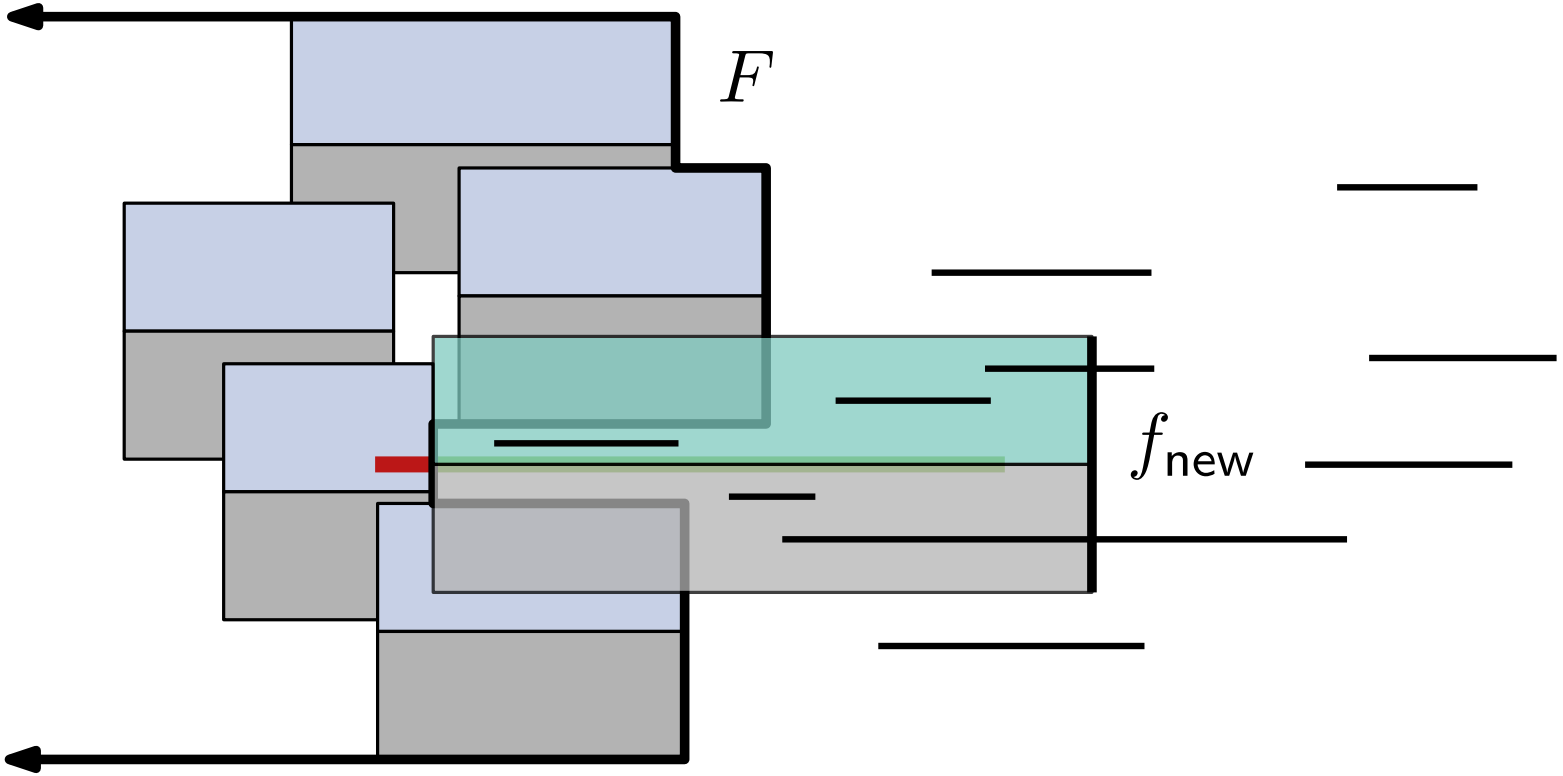
- speichere Obermenge V' mit $V_{\text{int,right}} \subseteq V'$
- min-Heap $\mathcal{H}_V$ der Menge V' geordnet nach x-Koordinaten
- falls Minimum in $\mathcal{H}_V$ nicht in $V_{\text{int,right}}$ liegt, verwerfe es und wähle nächstes Minimum

Das linkeste Label für den Greedy-Algorithmus ist eines der drei Minima von $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ und $\mathcal{H}_V$

Updates

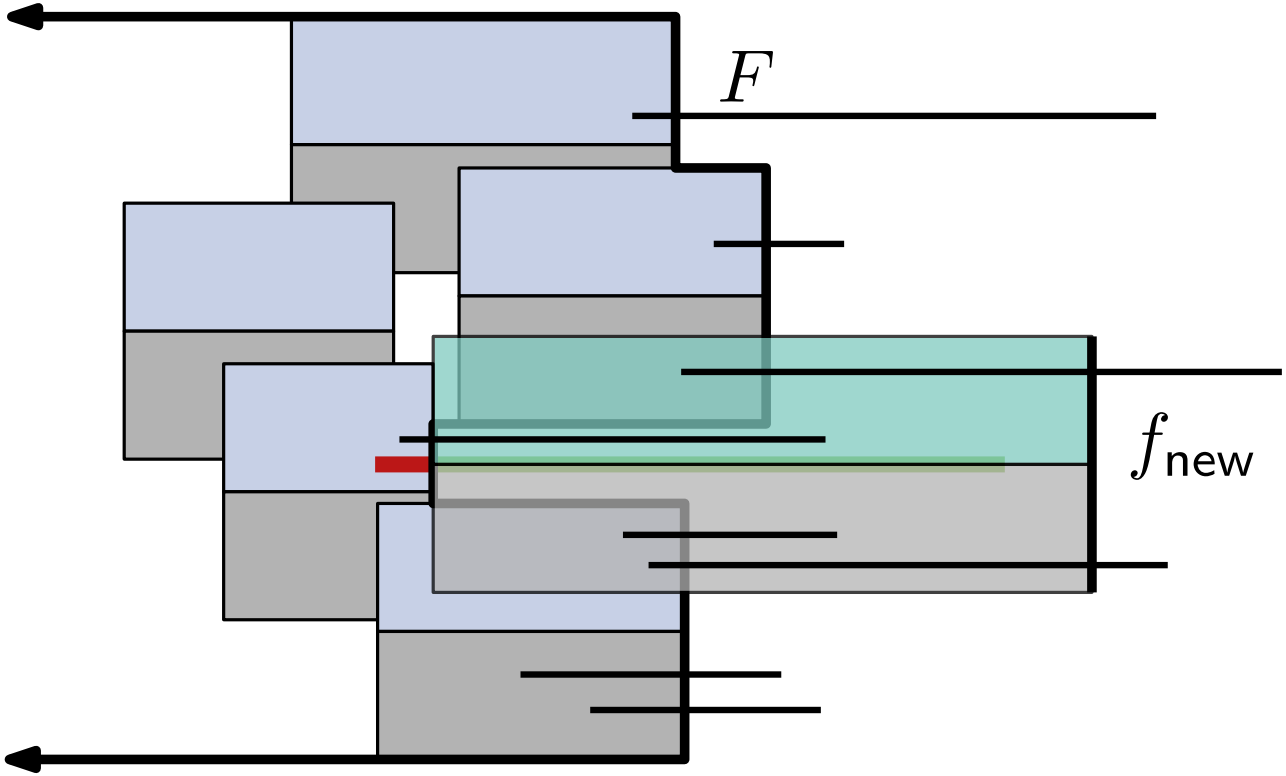


Updates



1) Updates von H_{right}

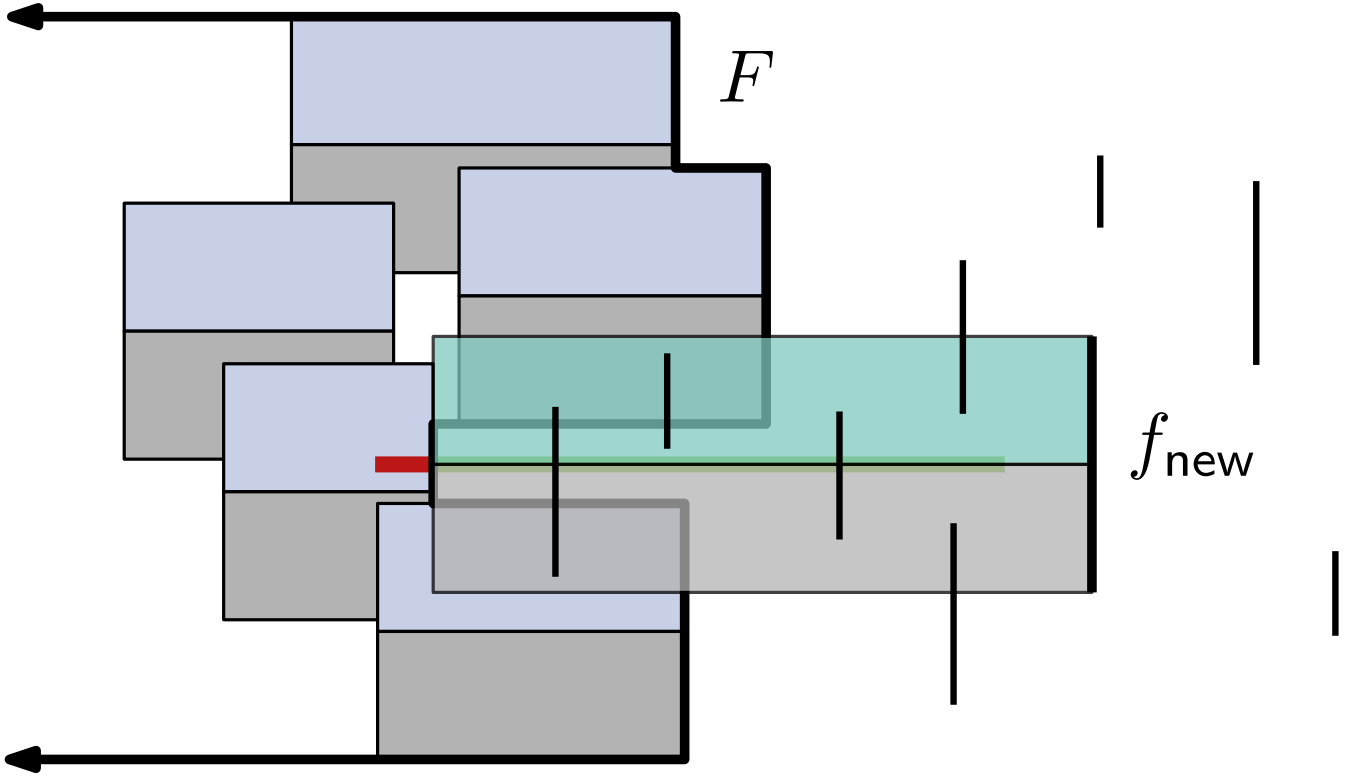
Updates



1) Updates von H_{right}

2) Updates von H_{int}

Updates



1) Updates von H_{right}
2) Updates von H_{int}
3) Updates von $V_{\text{int,right}}$ bzw. von V'

Algorithmus von van Kreveld, Strijk, Wolff

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

- $v \leftarrow$ find-min$(\mathcal{H}_V)$
- **while** v links von F **do**
  - delete-min$(\mathcal{H}_V)$; $v \leftarrow$ find-min$(\mathcal{H}_V)$
- $l_i \leftarrow$ 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 $\tilde{l_i}$
- update F und $\mathcal{T}_V$ mit f_{new}
- suche mit der Region links von f_{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

Algorithmus von van Kreveld, Strijk, Wolff

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

- $v \leftarrow$ find-min($\mathcal{H}_V$)
- **while** v links von F **do**
  - delete-min($\mathcal{H}_V$); $v \leftarrow$ find-min($\mathcal{H}_V$)
- $l_i \leftarrow$ 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 $\tilde{l_i}$
- update F und $\mathcal{T}_V$ mit f_{new}
- suche mit der Region links von f_{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

Satz 2: Der Algorithmus Greedy4S(P, L) lässt sich mit $O(n \log n)$ Zeitbedarf und $O(n)$ Platzbedarf implementieren. Die Lösung enthält mindestens halb so viele Label wie die optimale Lösung.

Ausblick

Van Kreveld, Strijk und Wolff geben auch ein polynomielles Approximationsschema (PTAS) für das Beschriften im Slider-Modell an.
Für Approximationsgüte $(1 - \varepsilon)$ ist die Laufzeit $O(n^{4/\varepsilon^2})$.

Ausblick

Van Kreveld, Strijk und Wolff geben auch ein polynomielles Approximationsschema (PTAS) für das Beschriften im Slider-Modell an.
Für Approximationsgüte $(1 - \varepsilon)$ ist die Laufzeit $O(n^{4/\varepsilon^2})$.

Ähnliche Ergebnisse (NP-vollständig, Approximation, PTAS) gelten auch für fixed-position-Modelle, z.B. (Agarwal et al. '98)

Ausblick

Van Kreveld, Strijk und Wolff geben auch ein polynomielles Approximationsschema (PTAS) für das Beschriften im Slider-Modell an.
Für Approximationsgüte $(1 - \varepsilon)$ ist die Laufzeit $O(n^{4/\varepsilon^2})$.

Ähnliche Ergebnisse (NP-vollständig, Approximation, PTAS) gelten auch für fixed-position-Modelle, z.B. (Agarwal et al. '98)

Slider-Modelle erlauben in realen Instanzen bis zu 15% mehr platzierte Labels als ein 4-Positionen-Modell.