Mehrdimensionale Block-Clusterbäume

Roman Jud, 30.04.2004

Vortragsübersicht

- Rückblende
  - *Erinnerung an die wesentlichen Begriffe*
- Hauptteil
  - *Definitionen*
  - *Zulässigkeitsbedingung*
  - *Bounding Boxes*
  - *Implementierung*
  - *abschliessendes Beispiel*

Rückblende

Definition: *Baum T (tree)*

Das tupel $T := (r, V, E, m, L)$ heisst (bezeichneter) Baum

wobei

r Wurzel (root)

V Menge der Eckpunkte/Knoten (vertices)

E Menge der Kanten (edges)

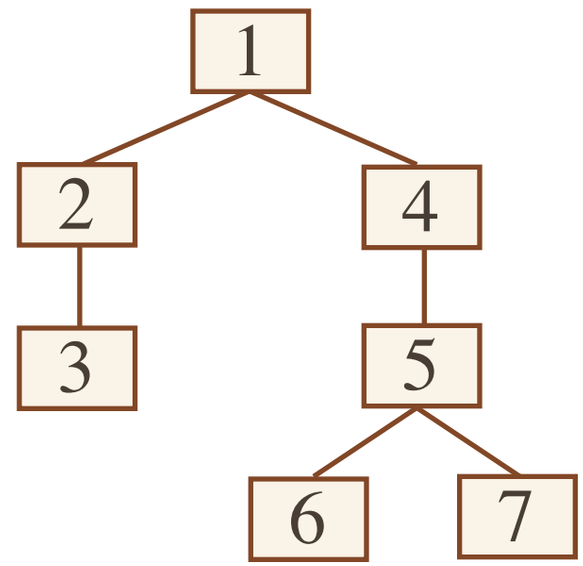
(*Menge höchstens zweielementiger Teilmengen von V*)

L Bezeichnermenge (labels)

$:\Longleftrightarrow$

- (V, E) ist verbundener, azyklischer Graph
- $r := \text{root}(T) \in V$
- $m : V \to L$ ist eine Abbildung

  $m(v) \in L$ heisst Bezeichnung von v

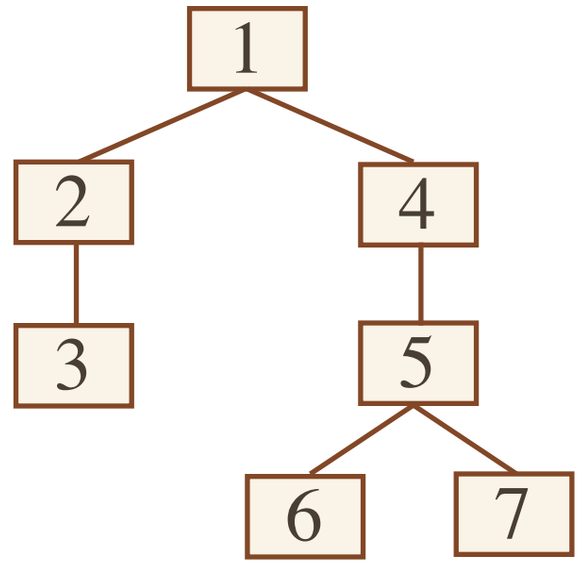


Notation:

$$\hat{v} := m(v)$$

Zu jedem Knoten $v \in V(T)$ gibt es genau einen Pfad $(v_i)_{i=0}^{\ell} \subset V^{\ell}$ mit

$$v_0 = \mathrm{root}(T) \quad \text{und} \quad v_\ell = v$$

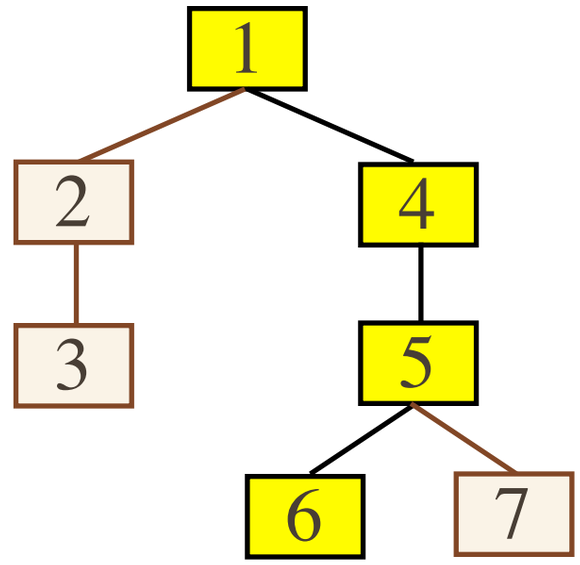


Zu jedem Knoten $v \in V(T)$ gibt es genau einen Pfad $(v_i)_{i=0}^{\ell} \subset V^{\ell}$ mit

$$v_0 = \mathrm{root}(T) \quad \text{und} \quad v_\ell = v$$

Daher folgende Bezeichnungen:

- $(v_i)_{i=0}^{\ell}$ heisst **Ahnenfolge von** v (sequence of ancestors)

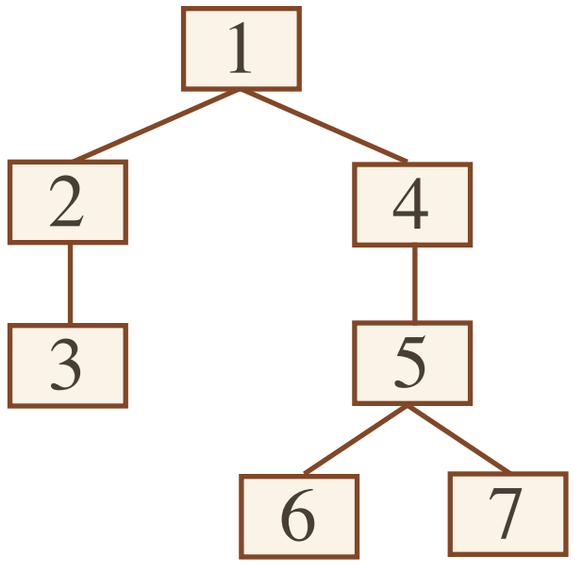


Zu jedem Knoten $v \in V(T)$ gibt es genau einen Pfad $(v_i)_{i=0}^{\ell} \subset V^{\ell}$ mit

$$v_0 = \mathrm{root}(T) \quad \text{und} \quad v_\ell = v$$

Daher folgende Bezeichnungen:

- $(v_i)_{i=0}^{\ell}$ heisst **Ahnenfolge von** v (sequence of ancestors)
- $\ell \in \mathbb{N}_0$ heisst **Ebene von** v; Notation: $\ell = \mathrm{level}(v)$
- $\mathrm{depth}(T) := \max\{\mathrm{level}(v) \,|\, v \in V(T)\}$
  heisst **Tiefe des Baums** T



Ebene 0

Ebene 1

Ebene 2

Ebene 3

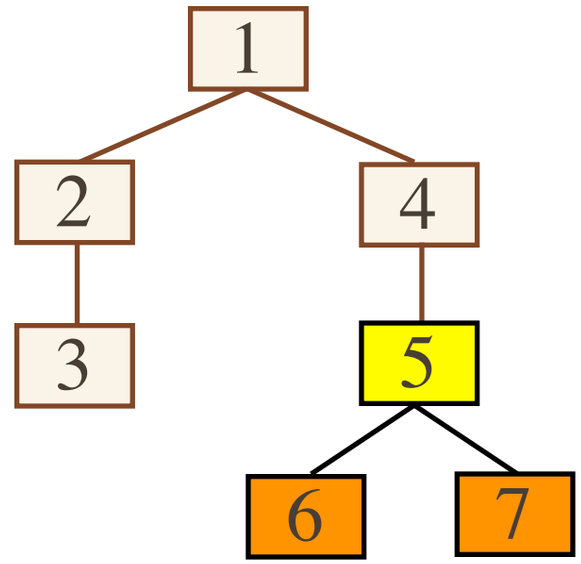
depth(T)=3

Zu jedem Knoten $v \in V(T)$ gibt es genau einen Pfad $(v_i)_{i=0}^{\ell} \subset V^{\ell}$ mit

$$v_0 = \mathrm{root}(T) \quad \text{und} \quad v_\ell = v$$

Daher folgende Bezeichnungen:

- $(v_i)_{i=0}^{\ell}$ heisst **Ahnenfolge von** v (sequence of ancestors)
- $\ell \in \mathbb{N}_0$ heisst **Ebene von** v; Notation: $\ell = \mathrm{level}(v)$
- $\mathrm{depth}(T) := \max\{\mathrm{level}(v) \,|\, v \in V(T)\}$
  heisst **Tiefe des Baums** T
- $v' \in V(t)$ heisst **Vater von** v; Notation: $v' = \mathrm{father}(v)$
  $$:\Leftrightarrow$$
  $$\ell > 0 \quad \text{und } v' = v_{\ell-1}$$
- $\mathrm{sons}(v) := \left\{ v' \in V(T) \backslash \{\mathrm{root}(T)\} \,|\, \mathrm{father}(v') = v \right\}$
  heisst **Menge der Söhne von** v

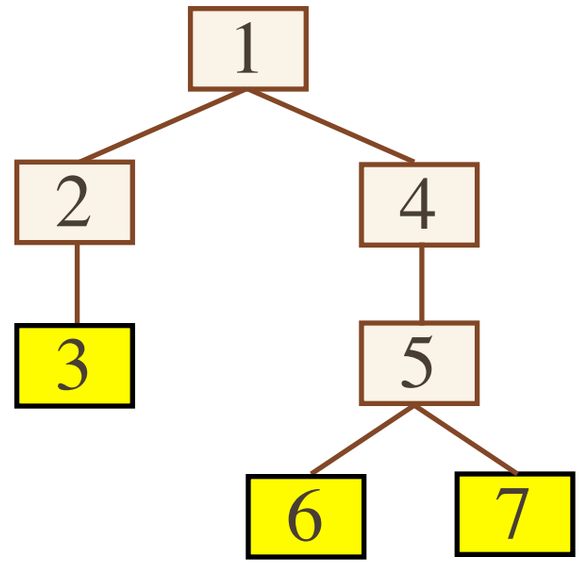


Zu jedem Knoten $v \in V(T)$ gibt es genau einen Pfad $(v_i)_{i=0}^{\ell} \subset V^{\ell}$ mit

$$v_0 = \mathrm{root}(T) \quad \text{und} \quad v_\ell = v$$

Daher folgende Bezeichnungen:

- $(v_i)_{i=0}^{\ell}$ heisst **Ahnenfolge von** v (sequence of ancestors)
- $\ell \in \mathbb{N}_0$ heisst **Ebene von** v; Notation: $\ell = \mathrm{level}(v)$
- $\mathrm{depth}(T) := \max\{\mathrm{level}(v) \,|\, v \in V(T)\}$
  heisst **Tiefe des Baums** T



- $v' \in V(t)$ heisst **Vater von** v; Notation: $v' = \mathrm{father}(v)$
  $$:\Leftrightarrow$$
  $$\ell > 0 \quad \text{und } v' = v_{\ell-1}$$
- $\mathrm{sons}(v) := \left\{v' \in V(T)\backslash\{\mathrm{root}(T)\} \,|\, \mathrm{father}(v') = v\right\}$
  heisst **Menge der Söhne von** v
- $v \in V(T)$ heisst **Blatt** $:\Leftrightarrow$ $\mathrm{sons}(v) = \emptyset$

Definition: *Clusterbaum* $T_{\mathcal{I}}$ *(cluster tree)*

Ein endlicher Baum (d.h. $\#V(T) < \infty$) heisst Clusterbaum für eine Indexmenge $\mathcal{I}$

$$:\Longleftrightarrow$$

- *"Wurzel-Bedingung"*

  $$\widehat{\text{root}(T)} = \mathcal{I}$$

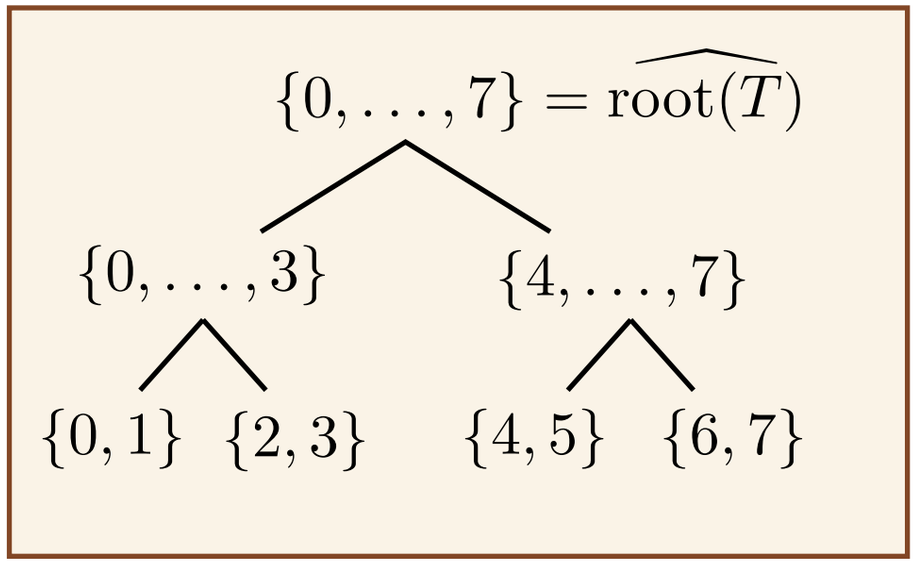
- *"Disjunktheits-Bedingung"*

  $\forall t \in V(T)$ gilt :

  $$s_1, s_2 \in \text{sons}(t), s_1 \neq s_2 \Rightarrow \widehat{s_1} \cap \widehat{s_2} = \emptyset$$

  und

  $$\text{sons}(t) \neq \emptyset \Rightarrow \hat{t} = \bigcup_{s \in \text{sons}(t)} \hat{s}$$



Definition: *Clusters*

Die Knoten $t \in V(T)$ *eines Clusterbaumes* $T_{\mathcal{I}} := T$ *heissen Clusters.*

Notation: $t \in T_{\mathcal{I}}$ anstatt $t \in V(T_{\mathcal{I}})$

Hauptteil

Definition: *Block-Clusterbaum* $T_{\mathcal{I}\times\mathcal{J}}$

Seien $T_{\mathcal{I}}$ und $T_{\mathcal{J}}$ Clusterbäume zu den Indexmengen $\mathcal{I}, \mathcal{J}$
Ein endlicher Baum $T := T_{\mathcal{I}\times\mathcal{J}}$ heisst Block-Clusterbaum für $T_{\mathcal{I}}$ und $T_{\mathcal{J}}$

$:\Longleftrightarrow$

- $\text{root}(T_{\mathcal{I}\times\mathcal{J}}) = \mathcal{I}\times\mathcal{J}$
- Jeder Knoten $b \in V(T_{\mathcal{I}\times\mathcal{J}})$ ist von der Form $b = (t, s)$
  für clusters $t \in T_{\mathcal{I}}, s \in T_{\mathcal{J}}$.

  Notation: $\boxed{b = t \times s}$ (Betonung des Zusammenhangs zwischen Indexmengen und den Clusters)

- Für alle Knoten $t \times s \in V(T_{\mathcal{I}\times\mathcal{J}})$ gelten folgende

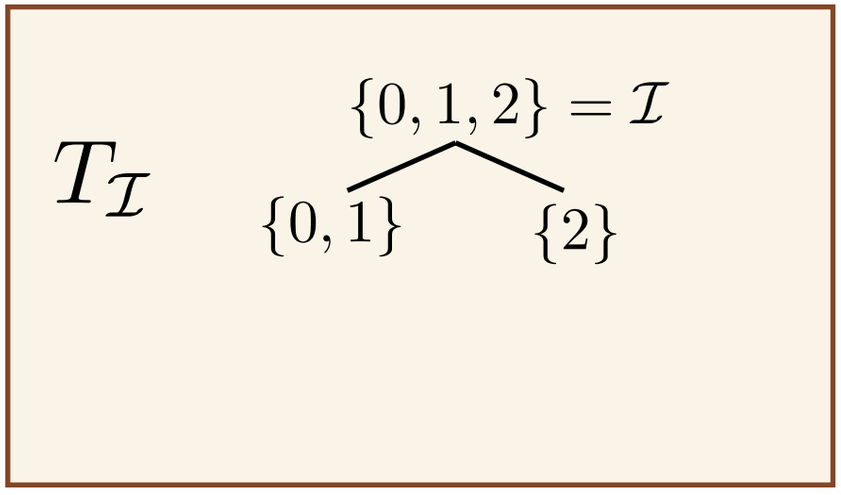
Bildungsregeln für die Menge der Söhne:

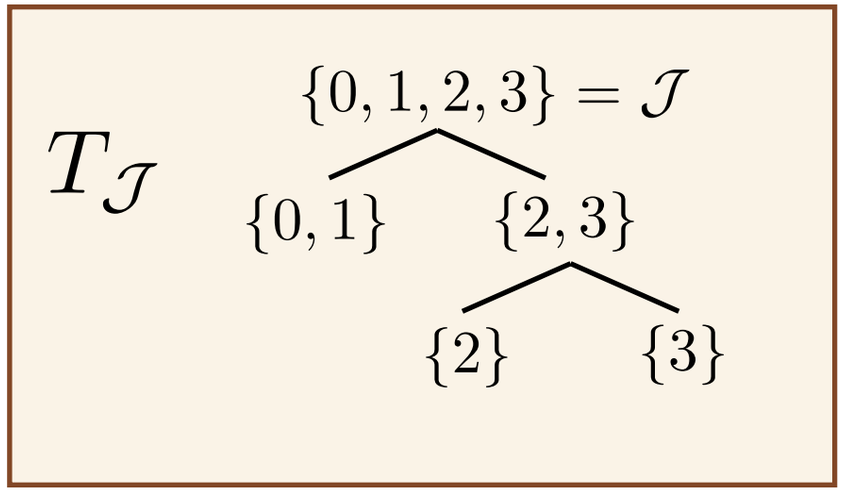
$$\text{sons}(t) = \emptyset \text{ und } \text{sons}(s) = \emptyset \Longrightarrow \text{sons}(t\times s) = \emptyset$$
$$\text{sons}(t) \neq \emptyset \text{ und } \text{sons}(s) = \emptyset \Longrightarrow \text{sons}(t\times s) = \{t'\times s \mid t' \in \text{sons}(t)\}$$
$$\text{sons}(t) = \emptyset \text{ und } \text{sons}(s) \neq \emptyset \Longrightarrow \text{sons}(t\times s) = \{t\times s' \mid s' \in \text{sons}(s)\}$$
$$\text{sons}(t) \neq \emptyset \text{ und } \text{sons}(s) \neq \emptyset \Longrightarrow \text{sons}(t\times s) = \{t'\times s' \mid t' \in \text{sons}(t), s' \in \text{sons}(s)\}$$

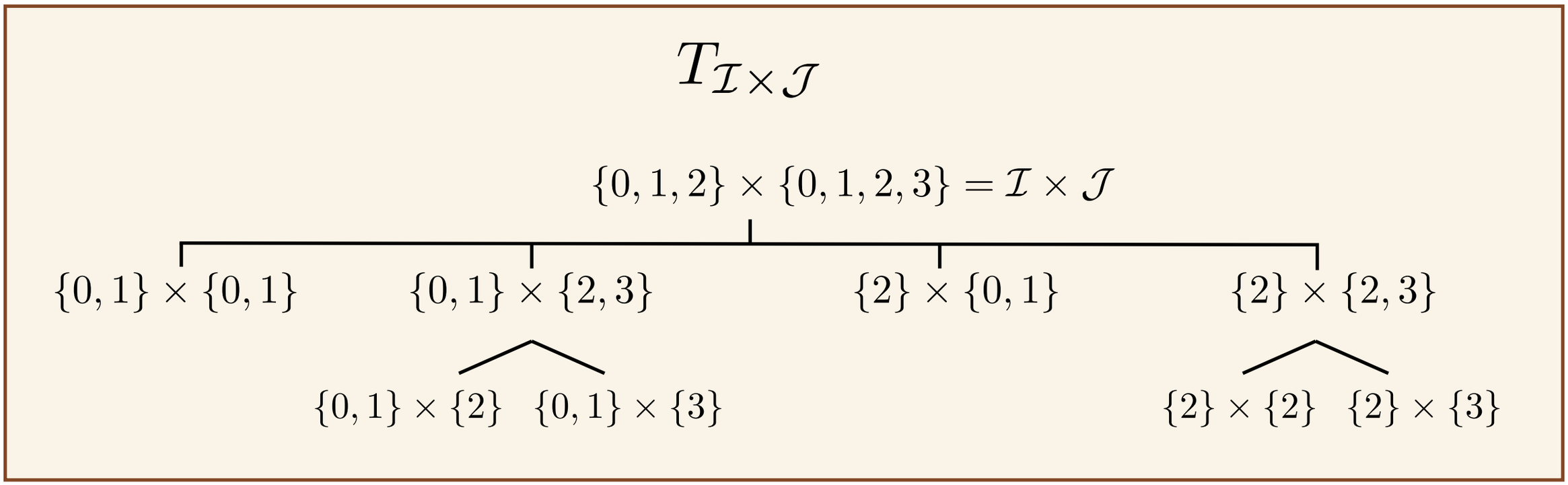
- Die Bezeichnung eines Knotens $t \times s \in V(T_{\mathcal{I}\times\mathcal{J}})$ ist gegeben durch

$$\widehat{t\times s} = \hat{t}\times\hat{s} \subset \mathcal{I}\times\mathcal{J}$$

Beispiel eines Block-Clusterbaums



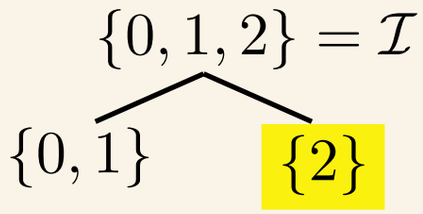




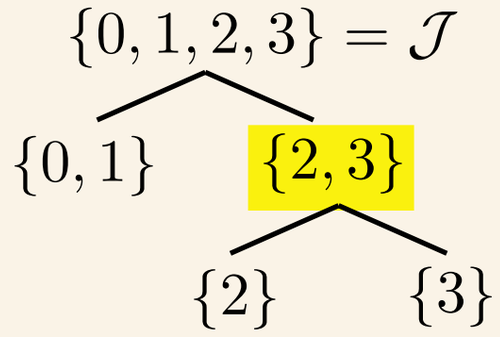
Regel:

$$\text{sons}(t) \neq \emptyset \text{ und } \text{sons}(s) \neq \emptyset \Longrightarrow \text{sons}(t \times s) = \{t' \times s' \mid t' \in \text{sons}(t), s' \in \text{sons}(s)\}$$

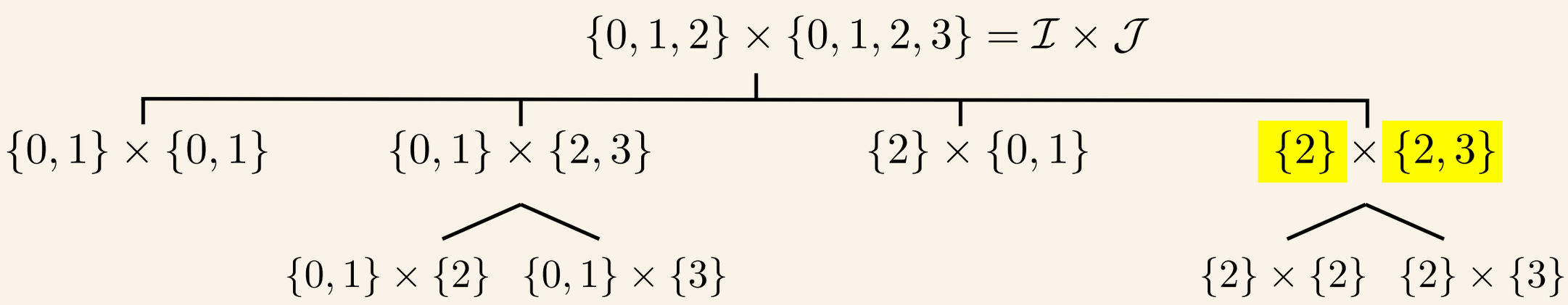
$T_{\mathcal{I}}$



$T_{\mathcal{J}}$



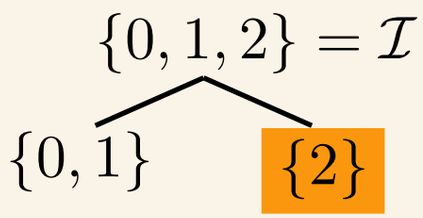
$T_{\mathcal{I} \times \mathcal{J}}$



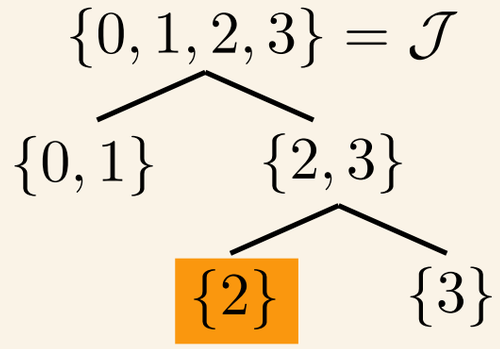
Regel:

$$\text{sons}(t) = \emptyset \text{ und } \text{sons}(s) \neq \emptyset \Longrightarrow \text{sons}(t \times s) = \{t \times s' \mid s' \in \text{sons}(s)\}$$

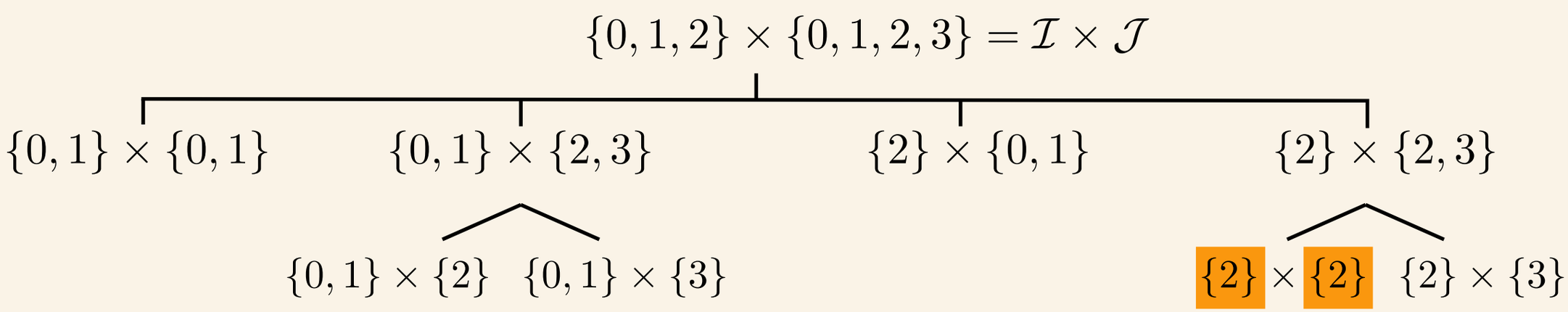
$T_{\mathcal{I}}$



$T_{\mathcal{J}}$



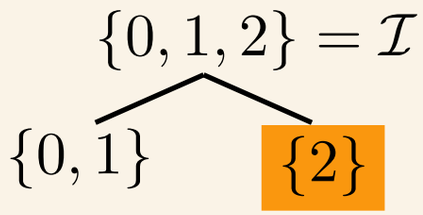
$T_{\mathcal{I}\times\mathcal{J}}$



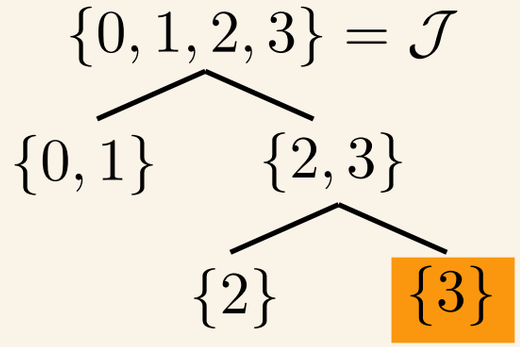
Regel:

$\text{sons}(t) = \emptyset$ und $\text{sons}(s) \neq \emptyset \Longrightarrow \text{sons}(t \times s) = \{t \times s' \mid s' \in \text{sons}(s)\}$

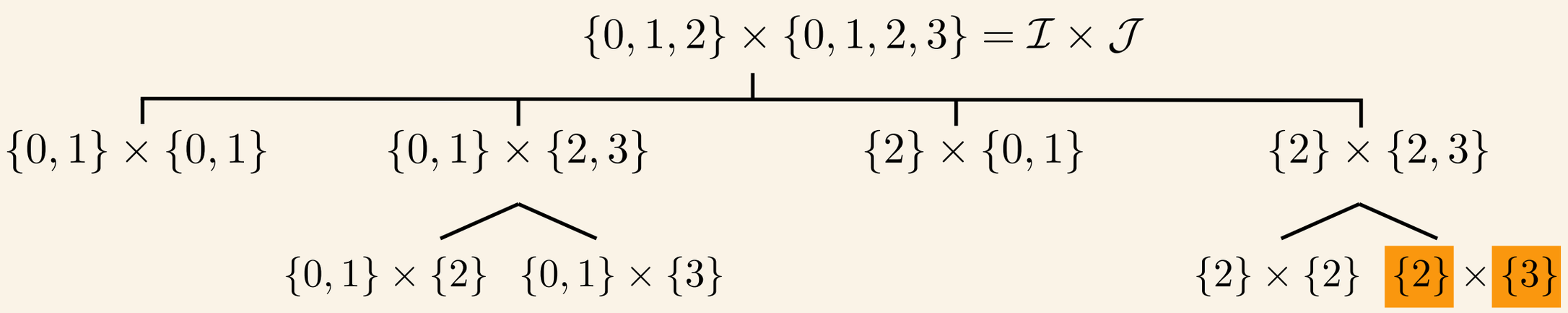
$T_{\mathcal{I}}$



$T_{\mathcal{J}}$

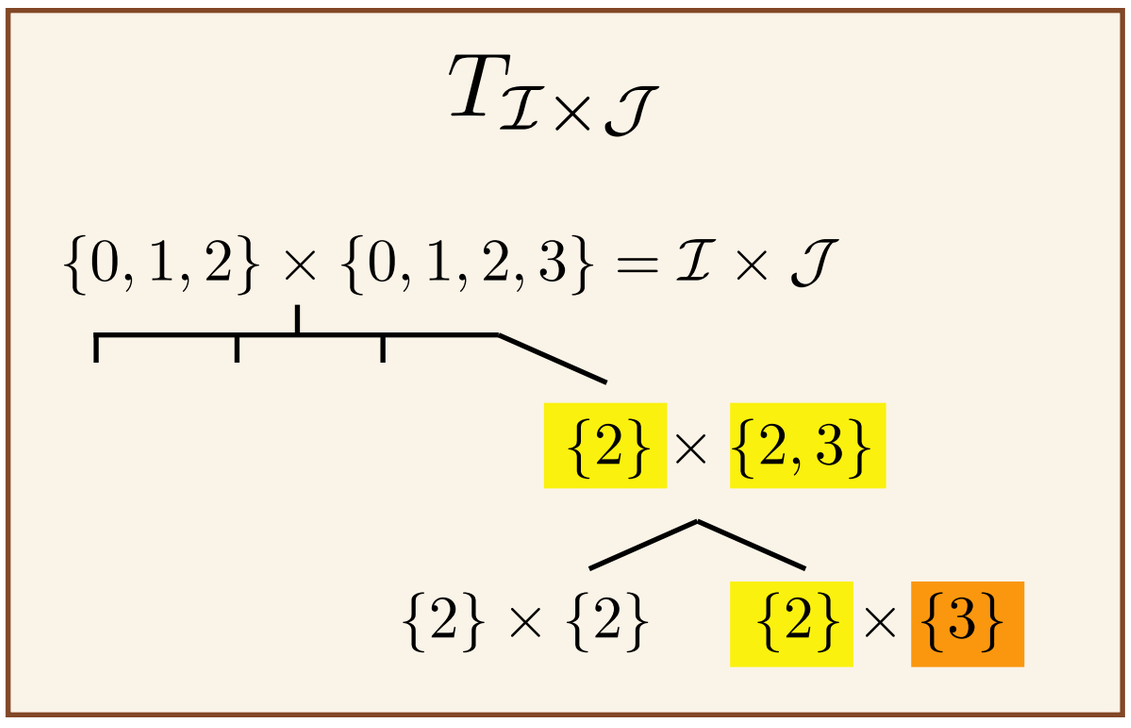


$T_{\mathcal{I} \times \mathcal{J}}$



Bemerkung:

Die Ebenen in den zum Block-Clusterbaum zugehörigen Clusterbäumen müssen nicht “homogen” sein.



Definition: *homogener Block-Clusterbaum*

Ein Block-Clusterbaum $T_{\mathcal{I}\times\mathcal{J}}$ zu $T_{\mathcal{I}}$ und $T_{\mathcal{J}}$ heisst **homogen**

$$:\Longleftrightarrow$$

Für alle Knoten $t \times s \in T_{\mathcal{I}\times\mathcal{J}}$ gilt:

$$\boxed{\operatorname{level}(t \times s) = \operatorname{level}(t) = \operatorname{level}(s)}$$

Haupteigenschaft homogener Block-Clusterbäume

Sei $T_{\mathcal{I}\times\mathcal{J}}$ ein ***homogener*** Block-Clusterbaum.

Für alle Knoten $t \times s \in T_{\mathcal{I}\times\mathcal{J}}$ gelten die Bildungsregeln:

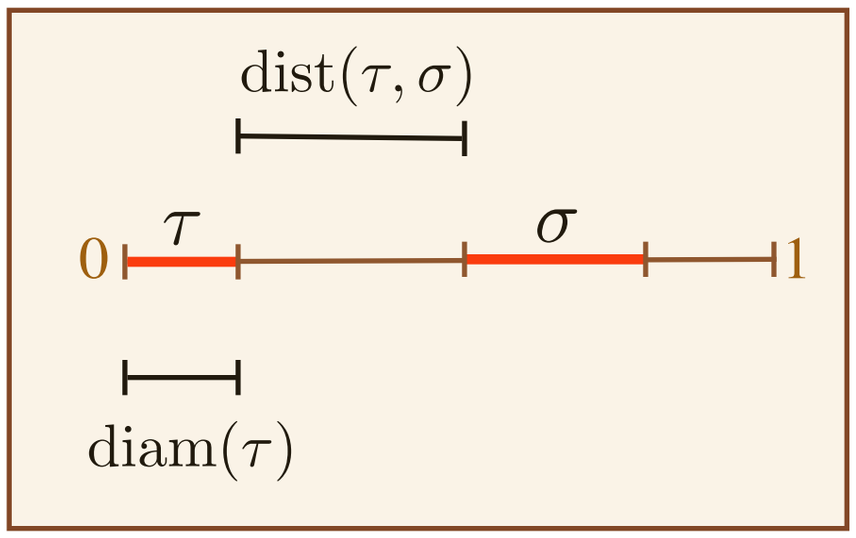
$$\begin{aligned}
&\operatorname{sons}(t) = \emptyset \text{ und } \operatorname{sons}(s) = \emptyset \Longrightarrow \operatorname{sons}(t \times s) = \emptyset \\
&\operatorname{sons}(t) \neq \emptyset \text{ und } \operatorname{sons}(s) = \emptyset \Longrightarrow \operatorname{sons}(t \times s) = \emptyset \\
&\operatorname{sons}(t) = \emptyset \text{ und } \operatorname{sons}(s) \neq \emptyset \Longrightarrow \operatorname{sons}(t \times s) = \emptyset \\
&\operatorname{sons}(t) \neq \emptyset \text{ und } \operatorname{sons}(s) \neq \emptyset \Longrightarrow \operatorname{sons}(t \times s) = \{t' \times s' \mid t' \in \operatorname{sons}(t), s' \in \operatorname{sons}(s)\}
\end{aligned}$$

Zulässigkeit

Erinnerung: *1-dim. Beispiel vom ersten Vortrag*

Zulässigkeitsbedingung (ZB)

$$\operatorname{diam}(\tau) \leq \operatorname{dist}(\tau, \sigma)$$



um zu entscheiden, ob wir eine Matrix durch Blöcke einer Niedrigrang-Matrix annähern können.

Für den mehrdimensionalen Fall müssen wir diese Zulässigkeitsbedingung verallgemeinern!

Mögliche Verallgemeinerung der ZB

- **Problem:** *Im mehrdimensionalen Fall entsprechen die Indizes in $\mathcal{I}$ nicht mehr den Intervallen.*
- **Abhilfe:** *Es gibt einen Zusammenhang zwischen Indizes und Definitionsbereichen, denn …*

Jeder Index $i \in \mathcal{I}$ gehört zu einer Basisfunktion φ_i und die Träger

$$\Omega_i := \operatorname{supp} \varphi_i$$

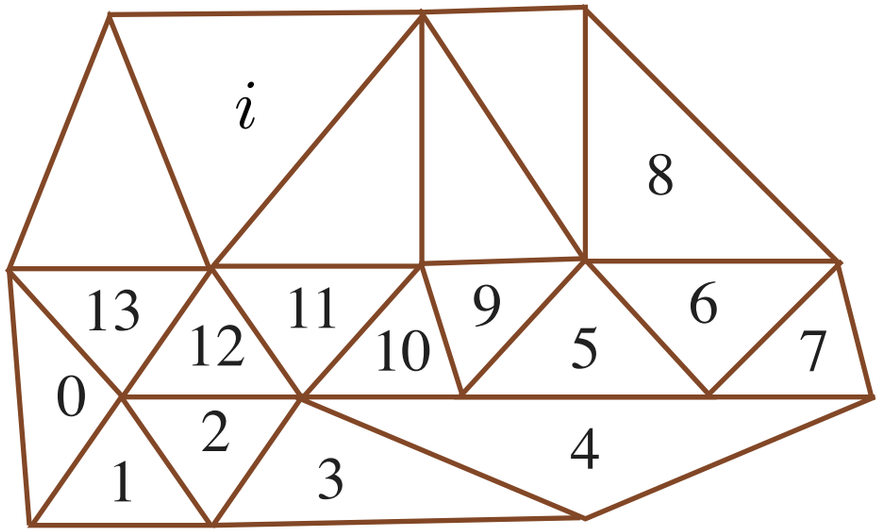
sind im d-dimensionalen Fall immer noch Teil-Definitionsbereiche des $\mathbb{R}^d$.
Wir setzen nun für ein Cluster $t \in T_{\mathcal{I}}$

$$\Omega_t := \bigcup_{i \in \hat{t}} \Omega_i$$

In Worten: Der zu $t \in T_{\mathcal{I}}$ gehörende Träger ist die minimale Teilmenge des $\mathbb{R}^d$, welche die Träger aller Basisfunktionen φ_i mit $i \in \hat{t}$ enthält.

Beispiel

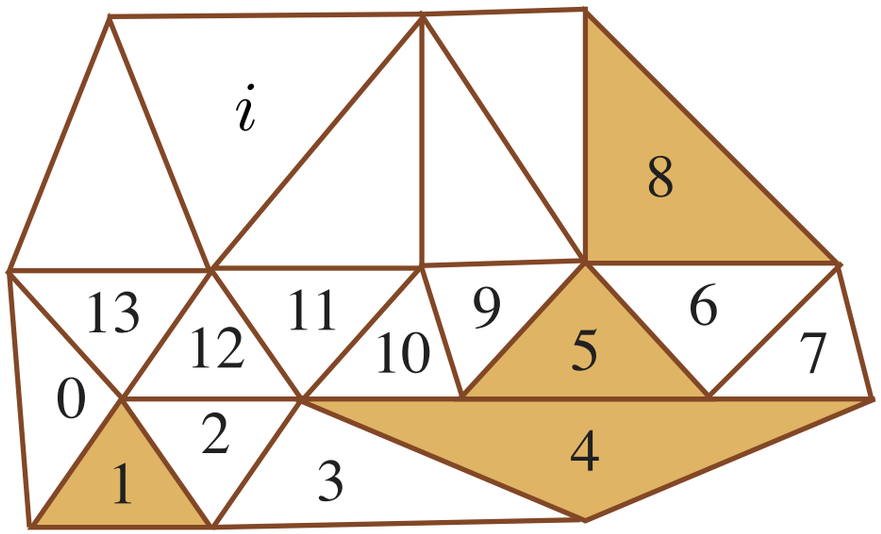
2D



$$\Omega_i := \operatorname{supp} \varphi_i = \triangle_i$$

Beispiel

2D



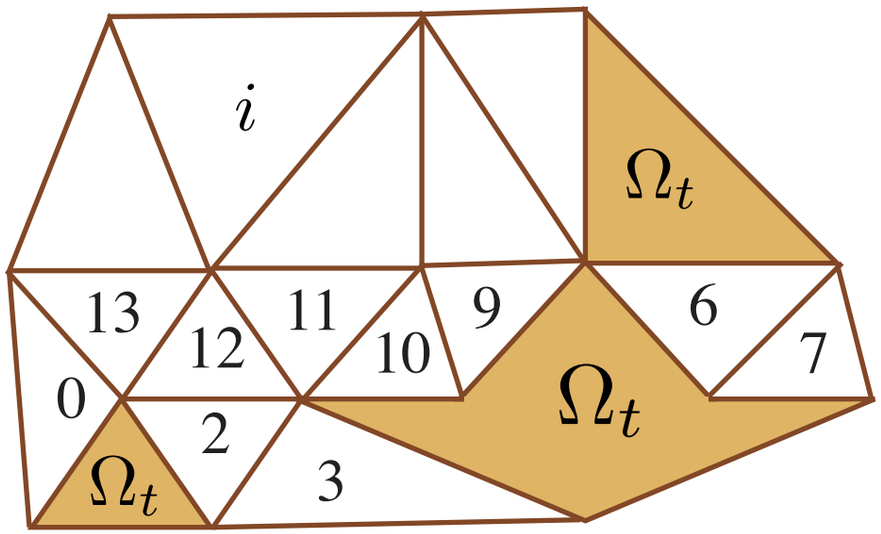
$$\Omega_i := \operatorname{supp} \varphi_i = \triangle_i$$

$$t = \{\triangle_1, \triangle_4, \triangle_5, \triangle_8\}$$

$$\hat{t} = \{1, 4, 5, 8\}$$

Beispiel

2D



$$\Omega_i := \operatorname{supp} \varphi_i = \triangle_i$$

$$t = \{\triangle_1, \triangle_4, \triangle_5, \triangle_8\}$$

$$\hat{t} = \{1, 4, 5, 8\}$$

also

$$\Omega_t := \bigcup_{i \in \hat{t}} \Omega_i = \bigcup_{i \in \{1,4,5,8\}} \triangle_i$$

Mögliche Verallgemeinerung der ZB

- **Problem:** *Im mehrdimensionalen Fall entsprechen die Indizes in $\mathcal{I}$ nicht mehr den Intervallen.*
- **Abhilfe:** *Es gibt einen Zusammenhang zwischen Indizes und Definitionsbereichen, denn …*

Jeder Index $i \in \mathcal{I}$ gehört zu einer Basisfunktion φ_i und die Träger

$$\Omega_i := \operatorname{supp} \phi_i$$

sind im d-dimensionalen Fall immer noch Teil-Definitionsbereiche des $\mathbb{R}^d$.
Wir setzen nun für ein Cluster $t \in T_{\mathcal{I}}$

$$\Omega_t := \bigcup_{i \in \hat{t}} \Omega_i$$

In Worten: Der zu $t \in T_{\mathcal{I}}$ gehörende Träger ist die minimale Teilmenge des $\mathbb{R}^d$, welche die Träger aller Basisfunktionen φ_i mit $i \in \hat{t}$ enthält.

Fazit

Eine mögliche Verallgemeinerung der ZB ist :

$$\boxed{\min\{\mathrm{diam}(\Omega_t), \mathrm{diam}\Omega_s\} \leq \eta \cdot \mathrm{dist}(\Omega_t, \Omega_s)} \quad \text{mit } \eta > 0$$

wobei

$$\mathrm{diam}(\Omega_t) = \sup\{\|x - y\| \mid x, y \in \Omega_t\}$$

$$\mathrm{dist}(\Omega_t, \Omega_s) = \inf\{\|x - y\| \mid x \in \Omega_t, y \in \Omega_s\}$$

mit dem euklidischen Abstand $\|\cdot\|$.

Zulässige Knoten eines Block-Clusterbums können nun folgendermassen definiert werden:

$t \times s \in T_{\mathcal{I}\times\mathcal{J}}$ heisst zulässig

$:\Leftrightarrow$

Ω_t und Ω_s sind zulässig.

Definition: *zulässiger Block-Clusterbaum*

Ein Block-Clusterbaum $T_{\mathcal{I}\times\mathcal{J}}$ für $\mathcal{I}$ und $\mathcal{J}$
heisst ***zulässig bezüglich einer Zulässigketsbedingung***

$$:\Longleftrightarrow$$

Für alle *Blätter* $t \times s \in \mathcal{L}(T_{\mathcal{I}\times\mathcal{J}})$ gilt eine der Bedingungen:

- $t \times s$ ist zulässig
- $\operatorname{sons}(t) = \emptyset$
- $\operatorname{sons}(s) = \emptyset$

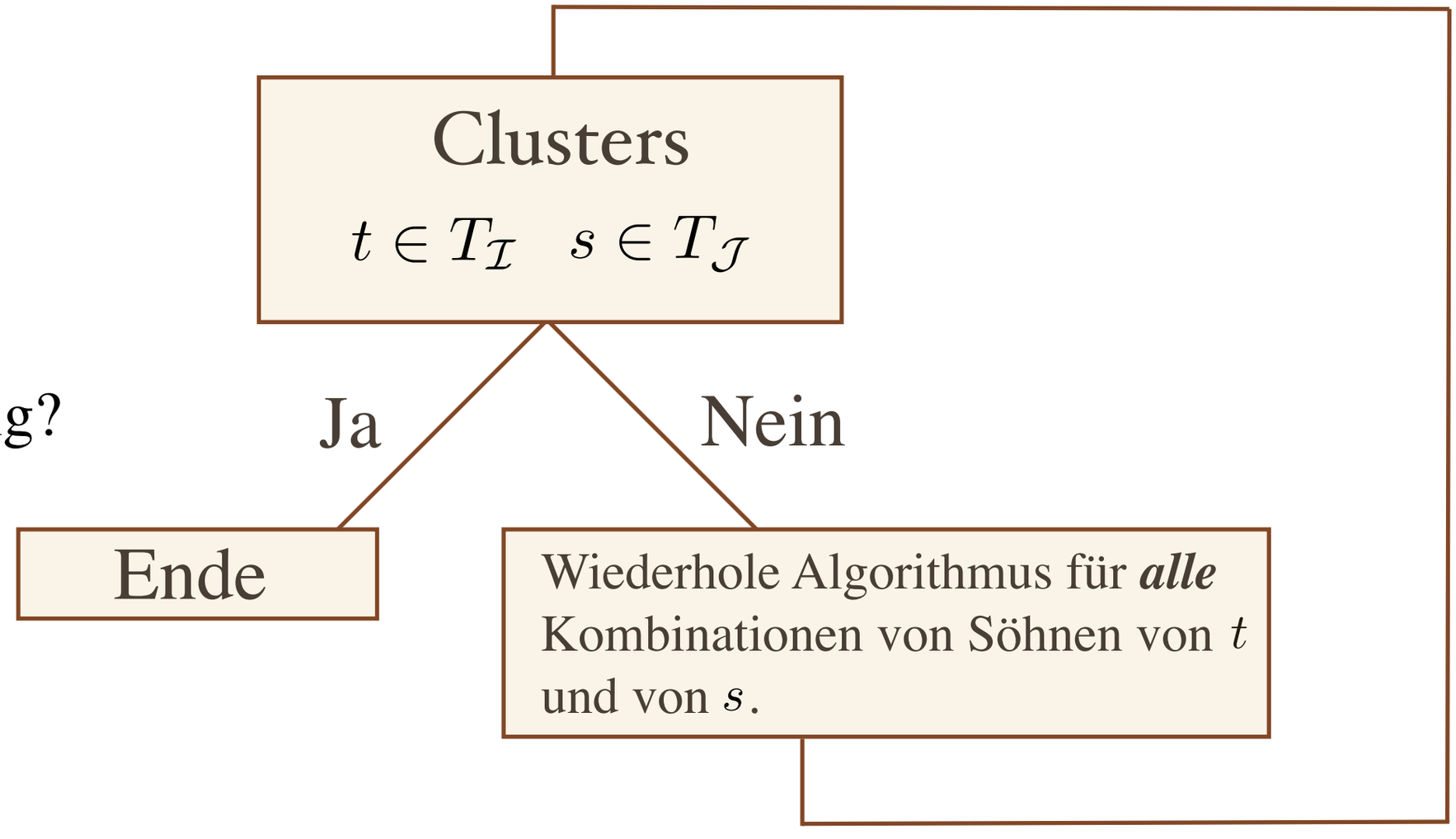
Konstruktionsprinzip für einen zulässigen Block-Clusterbaum

gegeben:

- zwei Clusterbäume $T_{\mathcal{I}}, T_{\mathcal{J}}$
- eine Zulässigkeitsbedingung

Eingabe

Ist $t \times s$ zulässig?



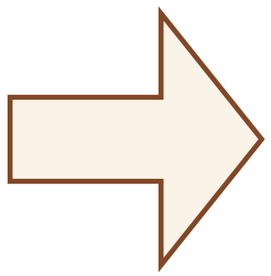
Bounding Boxes

- **Problem:** Das Überprüfen der Zulässigkeitsbedingung

$$\min\{\mathrm{diam}(\Omega_t), \mathrm{diam}\Omega_s\} \leq \eta \cdot \mathrm{dist}(\Omega_t, \Omega_s) \qquad \bigstar$$

ist **rechnerisch sehr aufwändig.**

- **Abhilfe:** Einfachere, strengere Bedingung, d.h. eine Bedingung, welche $\bigstar$ impliziert.



Eine Möglichkeit, eine solche Bedingung zu definieren, bieten die so genannten “bounding boxes”.

Definition: *achsenparallele Box*

Seien $a_1 < b_1, \ldots, a_d < b_d$.

$Q := [a_1, b_1] \times \cdots \times [a_d, b_d]$ heisst achsenparallele Box.

Definition: *Bounding Box des Clusters* t

Sei $t \in T_{\mathcal{I}}$ ein Cluster.

Eine achsenparallele Box $Q_t \subset \mathbb{R}^d$ heisst Bounding Box.

$$:\Longleftrightarrow$$

$$\boxed{\Omega_t \subset Q_t}$$

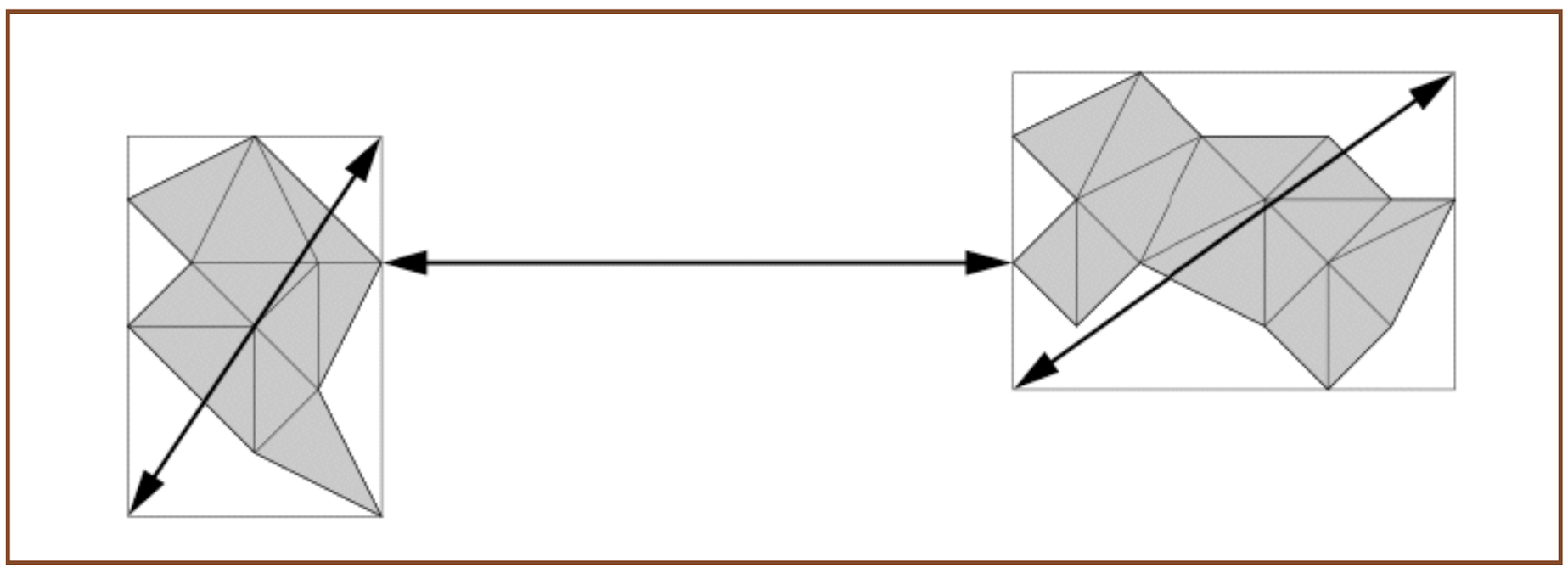
zur Erinnerung:

$$\Omega_t = \bigcup_{i \in \hat{t}} \Omega_i \quad \text{wobei} \quad \Omega_i = \operatorname{supp} \varphi_i$$

Fazit: Bounding-Box-Bedingung (BBB)

Die mit Hilfe der bounding boxes formulierte, einfachere Bedingung lautet nun:

$$\min\{\mathrm{diam}(Q_t), \mathrm{diam}(Q_s)\} \leq \eta \cdot \mathrm{dist}(Q_s, Q_t) \qquad \textbf{BBB}$$



Fazit: Bounding-Box-Bedingung (BBB)

Die mit Hilfe der bounding boxes formulierte, einfachere Bedingung lautet nun:

$$\min\{\mathrm{diam}(Q_t), \mathrm{diam}(Q_s)\} \leq \eta \cdot \mathrm{dist}(Q_s, Q_t) \qquad \textbf{BBB}$$

Wegen $\Omega_t \subset Q_t, \Omega_s \subset Q_s$ impliziert dies unsere verallg. ZB

$$\min\{\mathrm{diam}(\Omega_t), diam(\Omega_s)\} \leq \eta \cdot \mathrm{dist}(\Omega_t, \Omega_s)$$

d.h. **BBB** ist also eine einfachere und strengere ZB.

Wie überprüfen wir die BBB konkret?

gegeben:

$$Q_t := [a_1, b_1] \times \cdots \times [a_d, b_d]$$
$$Q_s := [c_1, d_1] \times \cdots \times [c_d, d_d]$$

Operationen:

$$\operatorname{diam}(Q_t) := \sqrt{\sum_{\ell=1}^{d} (b_\ell - a_\ell)^2} \qquad \operatorname{diam}(Q_s) := \sqrt{\sum_{\ell=1}^{d} (d_\ell - c_\ell)^2}$$

$$\operatorname{dist}(Q_t, Q_s) := \sqrt{\sum_{\ell=1}^{d} \operatorname{dist}([a_\ell, b_\ell], [c_\ell, d_\ell])^2}$$

Bemerkung: Diese Operationen generieren einen rechnerischen **Aufwand der Ordnung 1**, sind also relativ billig zu haben.

Implementierung

Zur Implementierung von bounding boxes müssen wir die im letzten Vortrag angelegten Datenstrukturen

`cluster` und `clusterfactory`

erweitern.

cluster: alte Version

```
typedef struct _cluster cluster;
typedef cluster* pcluster;

struct _cluster {
   int start;
   int size;

   int sons;
   pcluster* son;
};
```

`cluster`: alte Version

```
typedef struct _cluster cluster;
typedef cluster* pcluster;

struct _cluster {
   int start;
   int size;

   int sons;
   pcluster* son;
};
```

beschreiben $\hat{t}$:

`start` enthält den ersten Index von $\hat{t}$

`size` enthält die Anzahl Indizes

cluster: alte Version

```
typedef struct _cluster cluster;
typedef cluster* pcluster;

struct _cluster {
   int start;
   int size;

   int sons;
   pcluster* son;
};
```

sons enthält die Anzahl Söhne

`cluster`: alte Version

```
typedef struct _cluster cluster;
typedef cluster* pcluster;

struct _cluster {
   int start;
   int size;

   int sons;
   pcluster* son;
};
```

Das array **`son`** ist gefüllt mit den Zeigern auf die Söhne.

`cluster`: erweiterte Version

```
typedef struct _cluster cluster;
typedef cluster* pcluster;

struct _cluster {
   int start;
   int size;

   int sons;
   pcluster* son;

   double* bmin;
   double* bmax;
   int d;
};
```

Die Felder **`bmin`** und **`bmax`** sind d-dimensionale arrays vom typ `double`.

Sie enthalten die Minimal- und die Maximalkoordinaten, d.h.
falls

$$Q_t = [a_1, b_1] \times \cdots \times [a_d, b_d]$$

dann

$$\texttt{bmin} == (a_1, \ldots, a_d)$$
$$\texttt{bmax} == (b_1, \ldots, b_d)$$

Optimale Bounding Boxes (BB)

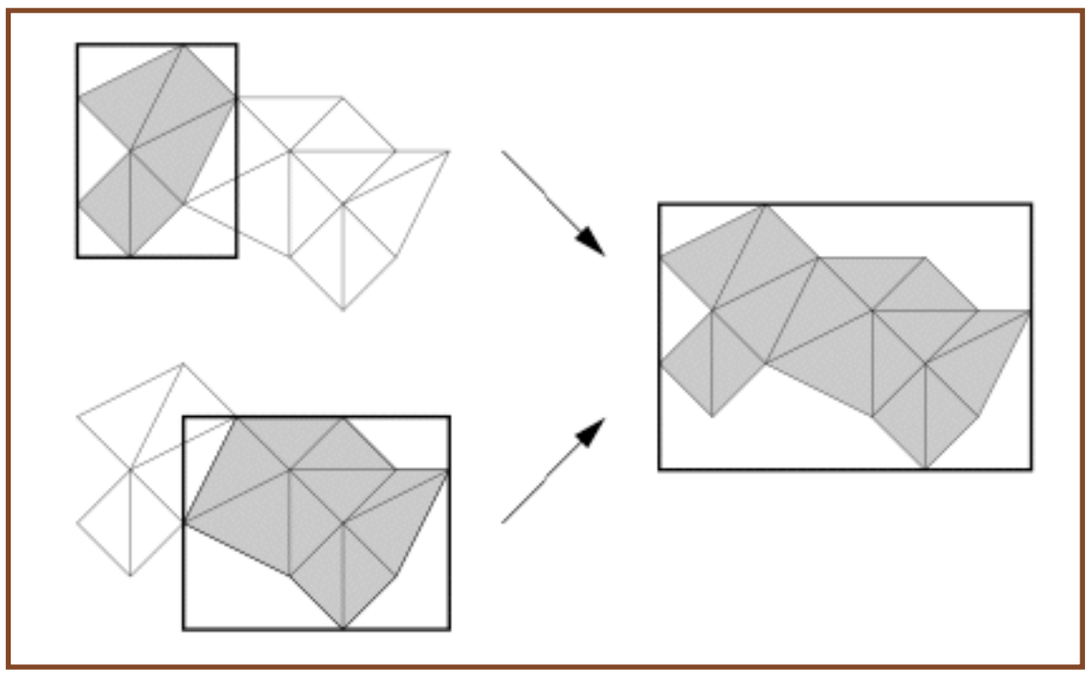
Die Struktur des Cluster-Baums hilft uns dabei, optimale Bounding Boxes zu konstruieren, d.h ***BB von minimalem Durchmesser****.*

Idee: Nach Def. gilt: $\hat{t} = \bigcup_{t' \in \mathrm{sons}(t)} \hat{t'}$. Daher folgende

Zerlegung $$\Omega_t = \bigcup_{t' \in \mathrm{sons}(t)} \Omega_{t'}$$

Die optimale BB zum Cluster t muss also die optimalen BB der Söhne t' enthalten.

Fazit: Die optimale BB kann konstruiert werden, indem wir die **Maxima und Minima der entsprechenden Koordinatenvektoren** bestimmen.



clusterfactory: alte Version

```
typedef struct _clusterfactory clustervactory;
typedef clusterfactory *pclusterfactory;

struct _clusterfactory {
   double **x;

   int nmax;
   int nn;
   int d;

   double *vmin;
   double *vmax;

   double *blocks;
};
```

clusterfactory: alte Version

```
typedef struct _clusterfactory clustervactory;
typedef clusterfactory *pclusterfactory;

struct _clusterfactory {
   double **x;

   int nmax;
   int nn;
   int d;

   double *vmin;
   double *vmax;

   double *blocks;
};
```

nmax	enthält die Maximalzahl der Freiheitsgrade
nn	enthält die Maximalzahl der Punkte
d	räumliche Dimension

`clusterfactory`: alte Version

```
typedef struct _clusterfactory clustervactory;
typedef clusterfactory *pclusterfactory;

struct _clusterfactory {
   double **x;

   int nmax;
   int nn;
   int d;

   double *vmin;
   double *vmax;

   double *blocks;
};
```

array **x** speichert die Koordinaten der Punkte x_i für jeden Index $i \in \{1, \ldots, nn-1\}$

Der Eintrag `x[i]` ist ein d-dimensionales array, welches die Kandidaten des Punktes x_i enthält.

clusterfactory: alte Version

```
typedef struct _clusterfactory clustervactory;
typedef clusterfactory *pclusterfactory;

struct _clusterfactory {
   double **x;

   int nmax;
   int nn;
   int d;

   double *vmin;
   double *vmax;

   double *blocks;
};
```

vmin und **vmax** werden gebraucht, um die Minimal- / Maximalwerte zu jeder Koordinate zu speichern.

`clusterfactory`: erweitert Version

```
typedef struct _clusterfactory clustervactory;
typedef clusterfactory *pclusterfactory;

struct _clusterfactory {
   double **x;

   int nmax;
   int nn;
   int d;

   double *vmin;
   double *vmax;

   double *blocks;

   double **smin;
   double **smax;
};
```

Die arrays **`smin`** und **`smax`** haben nn Einträge; für jeden Index einen.

Die Einträge `smin[i]` und `smax[i]` beschreiben die achsenparallele box, welche den Träger Ω_i der entsprechenden Basisfunktion φ_i enthält in der gleichen Weise, wie dies `bmin` und `bmax` für das cluster tun.

abschliessendes Beispiel

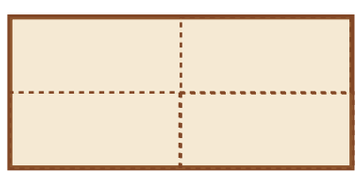
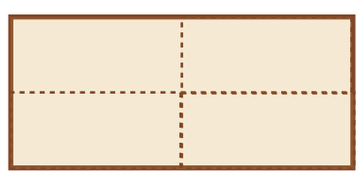
Erinnerung: $G_{ij} := \int\int \varphi_i(x) g(x,y) \varphi_j(y)\, dxdy$

mit Funktionenbasis $V_n := \{\varphi_0, \ldots, \varphi_{n-1}\}$

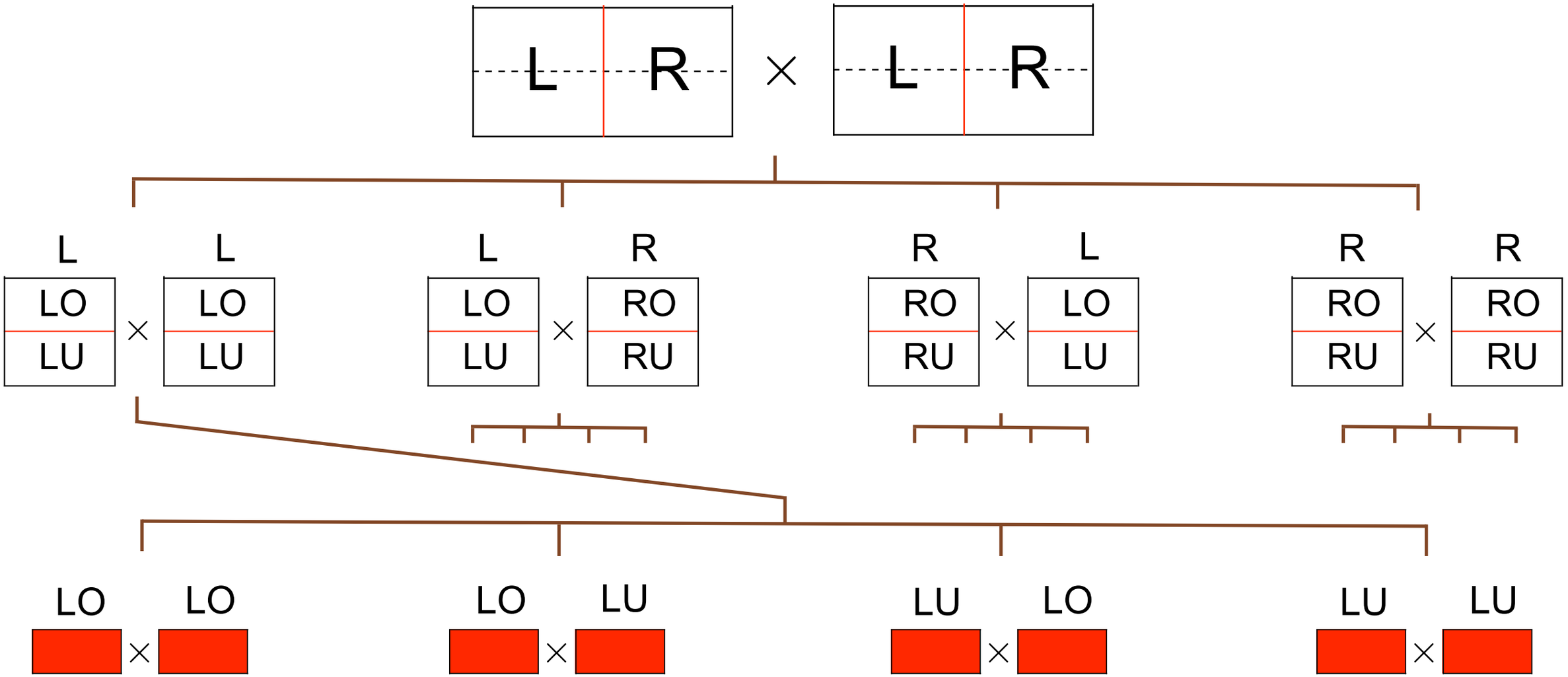
Bemerkungen:

- Für den x-Bereich und den y-Bereich haben wir dieselben Basisfunktionen, d.h. die Träger partitionieren den x-Bereich und den y-Bereich in gleicher Weise.
- Ziel ist es, möglichst grosse Cluster zu konstruieren, in denen wir G_{ij} mit Hilfe eines degenerierten Kerns $\tilde{g}(x,y) := \sum_{\nu} g_\nu(x) h_\nu(y)$ annähern können.

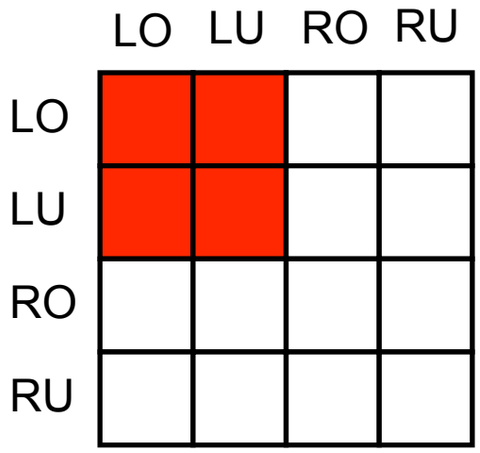
gegeben:

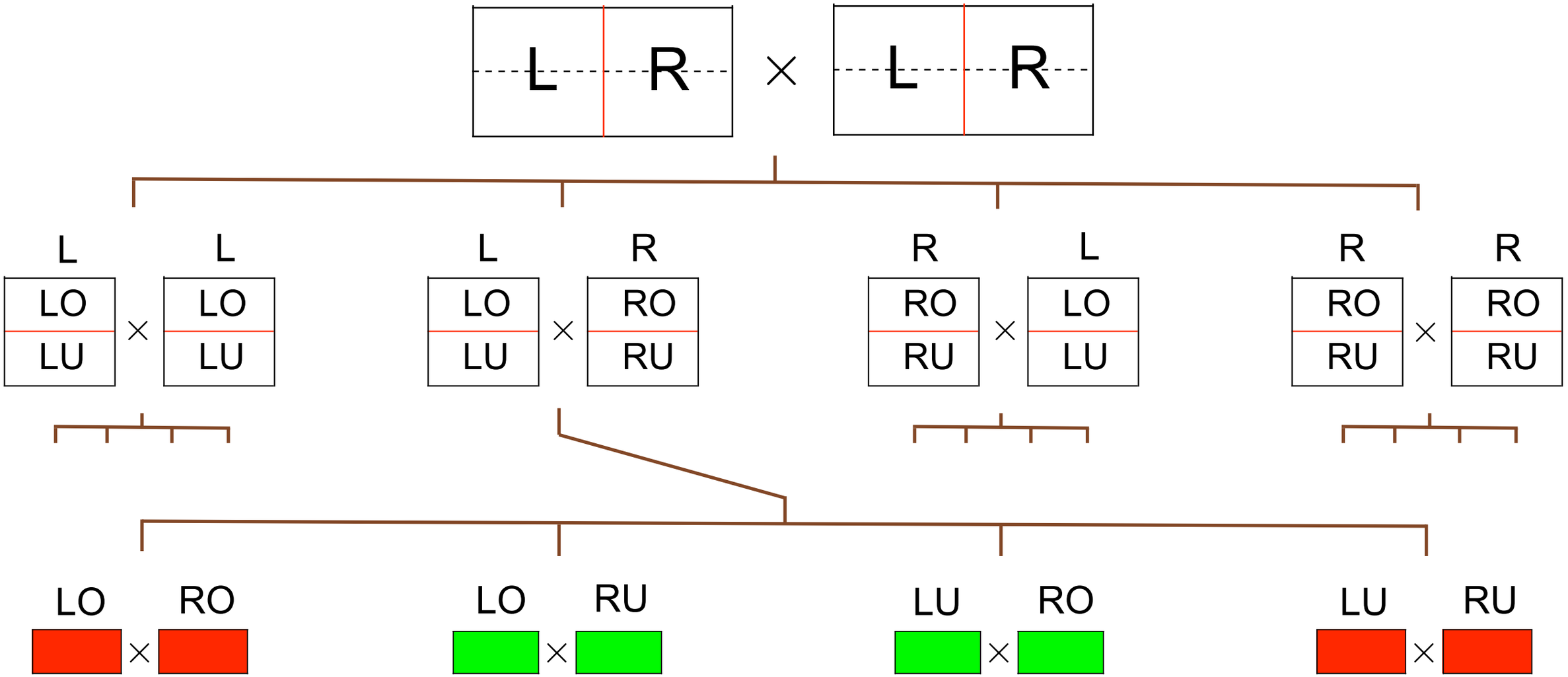
- x-Bereich  $\subset \mathbb{R}^2$ y-Bereich  $\subset \mathbb{R}^2$
- **Zulässigkeitsbedingung:**

  Zwei Cluster sind zulässig genau dann, wenn sie **sich in höchstens einem Punkt überschneiden**.

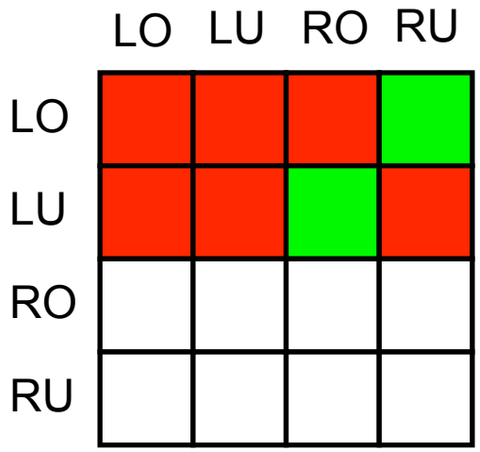

L
R
×
L
R
L
L
LO
LU
×
LO
LU
L
R
LO
LU
×
RO
RU
R
L
RO
RU
×
LO
LU
R
R
RO
RU
×
RO
RU
LO
LO
×
LO
LU
×
LU
LO
×
LU
LU
×

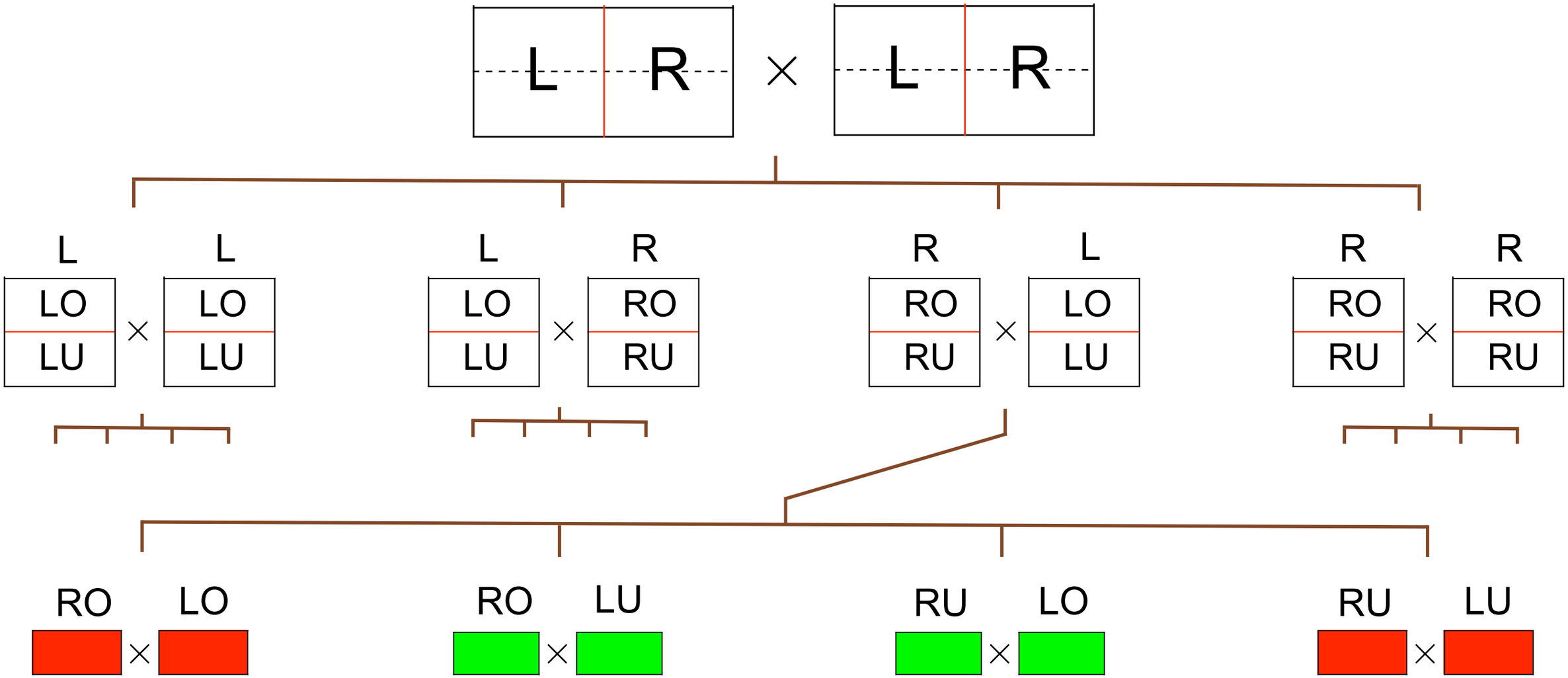
Matrix


LO
LU
RO
RU
LO
LU
RO
RU

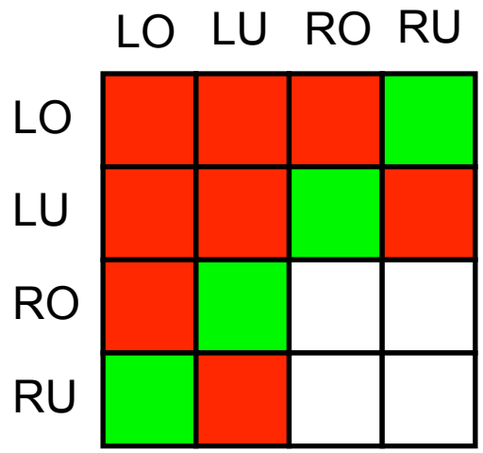

L
R
×
L
R
L
L
LO
LU
×
LO
LU
L
R
LO
LU
×
RO
RU
R
L
RO
RU
×
LO
LU
R
R
RO
RU
×
RO
RU
LO
RO
×
LO
RU
×
LU
RO
×
LU
RU
×

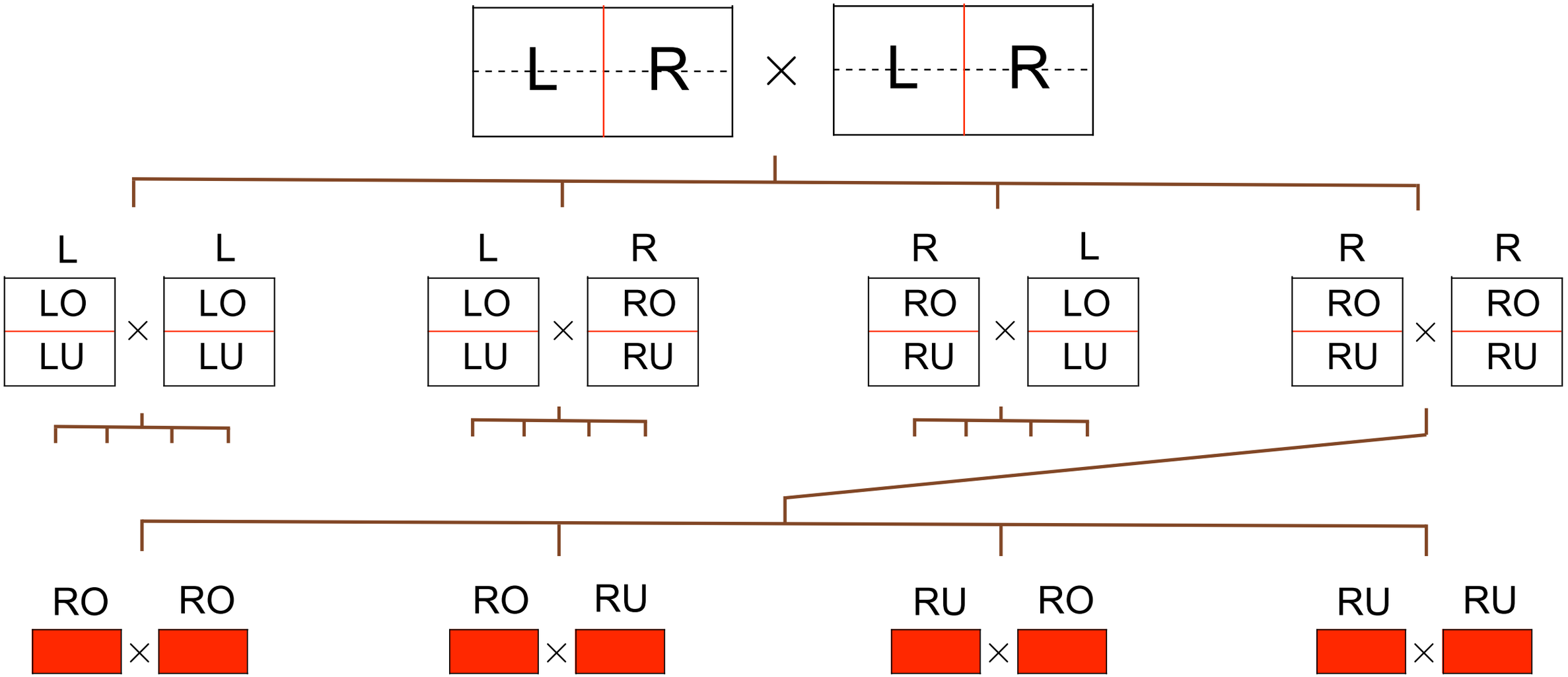
Matrix


LO
LU
RO
RU
LO
LU
RO
RU

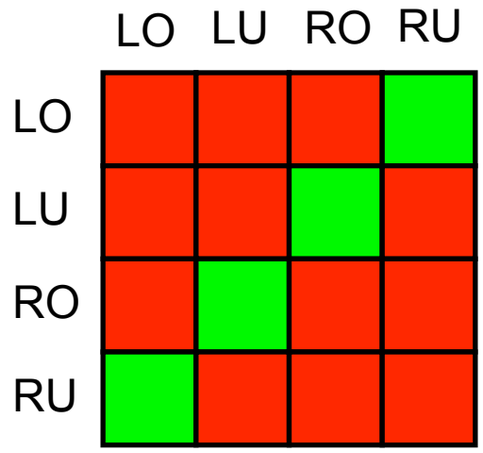

L
R
L
R
L
L
LO
LU
LO
LU
L
R
LO
LU
RO
RU
R
L
RO
RU
LO
LU
R
R
RO
RU
RO
RU
RO
LO
RO
LU
RU
LO
RU
LU

Matrix


LO
LU
RO
RU
LO
LU
RO
RU


L
R
L
R
LO
LU
LO
LU
L
L
LO
LU
RO
RU
R
L
RO
RU
LO
LU
R
R
RO
RU
RO
RU
RO
RO
RO
RU
RU
RO
RU
RU

Matrix


LO
LU
RO
RU
LO
LU
RO
RU

THE END

to be continued ...