Instance-optimal Database Joins

Mahmoud Abo Khamis
Relational AI

Logic and Algorithms in Database Theory and AI Boot Camp
Simons Institute

Aug 21, 2023

Table of Contents

Table of Contents

Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic

Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size N is no longer a lower bound on runtime

Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size N is no longer a lower bound on runtime
- Two stages

Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size N is no longer a lower bound on runtime
- Two stages
  - Preprocessing

Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size N is no longer a lower bound on runtime
- Two stages
  - Preprocessing
    - Sort input relations

Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size N is no longer a lower bound on runtime
- Two stages
  - Preprocessing
    - Sort input relations
    - Build indices, DS, etc

Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size N is no longer a lower bound on runtime
- Two stages
  - Preprocessing
    - Sort input relations
    - Build indices, DS, etc
  - Query Evaluation

Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size N is no longer a lower bound on runtime
- Two stages
  - Preprocessing
    - Sort input relations
    - Build indices, DS, etc
  - Query Evaluation
    - Reuse Prebuilt DSs for many queries (amortization)

Beyond Worst-case Analysis in DB: Why?

- Worst-case can be too pessimistic
- Input size N is no longer a lower bound on runtime
- Two stages
  - Preprocessing
    - Sort input relations
    - Build indices, DS, etc
  - Query Evaluation
    - Reuse Prebuilt DSs for many queries (amortization)
    - Sublinear time is possible

Beyond Worst-case Analysis: Some Models

- Parameterized Complexity
- Adaptive Analysis
- Instance Optimality
- Average-case
- ...

Instance Optimality: Goal

- Given an input instance I to some problem P

Instance Optimality: Goal

- Given an input instance I to some problem P
  - Find a lower bound $f(I)$ on the runtime of *any* algorithm A on I

Instance Optimality: Goal

- Given an input instance I to some problem P
  - Find a lower bound $f(I)$ on the runtime of *any* algorithm A on I
- Design an algorithm A^* whose runtime is $O(m \cdot f(I))$ for every I

Instance Optimality: Goal

- Given an input instance I to some problem P
  - Find a lower bound $f(I)$ on the runtime of *any* algorithm A on I
- Design an algorithm A^* whose runtime is $O(m \cdot f(I))$ for every I
  - m is the optimality ratio

Instance Optimality: General Approach

- Every algorithm must produce a proof $\mathcal{C}$ of output correctness (certificate)

Instance Optimality: General Approach

- Every algorithm must produce a proof $\mathcal{C}$ of output correctness (certificate)
- The minimum certificate size $|\mathcal{C}|$ is a lower bound on the runtime

Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS'03: Database aggregation problem

Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS'03: Database aggregation problem
- Meta-algorithm

Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS'03: Database aggregation problem
- Meta-algorithm
  - $\mathcal{C} \leftarrow \emptyset$ (The certificate)

Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS'03: Database aggregation problem
- Meta-algorithm
  - $\mathcal{C} \leftarrow \emptyset$ (The certificate)
  - While $\mathcal{C}$ does not yet prove the output

Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS'03: Database aggregation problem
- Meta-algorithm
  - $\mathcal{C} \leftarrow \emptyset$ (The certificate)
  - While $\mathcal{C}$ does not yet prove the output
    - $Q \leftarrow$ Some query to the input

Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS'03: Database aggregation problem
- Meta-algorithm
  - $\mathcal{C} \leftarrow \emptyset$ (The certificate)
  - While $\mathcal{C}$ does not yet prove the output
    - $Q \leftarrow$ Some query to the input
    - $\mathcal{C} \leftarrow \mathcal{C} \cup Q$

Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS'03: Database aggregation problem
- Meta-algorithm
  - $\mathcal{C} \leftarrow \emptyset$ (The certificate)
  - While $\mathcal{C}$ does not yet prove the output
    - $Q \leftarrow$ Some query to the input
    - $\mathcal{C} \leftarrow \mathcal{C} \cup Q$
    - Show that every certificate $\mathcal{C}'$ contains $\geqslant 1/m$ of Q

Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS'03: Database aggregation problem
- Meta-algorithm
  - $\mathcal{C} \leftarrow \emptyset$ (The certificate)
  - While $\mathcal{C}$ does not yet prove the output
    - $Q \leftarrow$ Some query to the input
    - $\mathcal{C} \leftarrow \mathcal{C} \cup Q$
    - Show that every certificate $\mathcal{C}'$ contains $\geqslant 1/m$ of Q
- Analysis

Instance Optimality: A Meta-Algorithm

- Fagin et al, JCSS'03: Database aggregation problem
- Meta-algorithm
  - $\mathcal{C} \leftarrow \emptyset$ (The certificate)
  - While $\mathcal{C}$ does not yet prove the output
    - $Q \leftarrow$ Some query to the input
    - $\mathcal{C} \leftarrow \mathcal{C} \cup Q$
    - Show that every certificate $\mathcal{C}'$ contains $\geqslant 1/m$ of Q
- Analysis
  - $|\mathcal{C}| \leqslant m \cdot |\mathcal{C}'|,$ for any certificate $\mathcal{C}'$

Table of Contents

Instance-optimal Set Intersection

- Input: Two sets R and S of numbers

Instance-optimal Set Intersection

- Input: Two sets R and S of numbers
- Output: $Q := R \cap S$

Instance-optimal Set Intersection

- Input: Two sets R and S of numbers
- Output: $Q := R \cap S$
  - $Q(X) = R(X) \wedge S(X)$

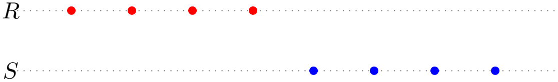
Instance-optimal Set Intersection

- Input: Two sets R and S of numbers
- Output: $Q := R \cap S$
  - $Q(X) = R(X) \wedge S(X)$
- Worst-case Runtime: $O(\min(|R|, |S|))$

Instance-optimal Set Intersection

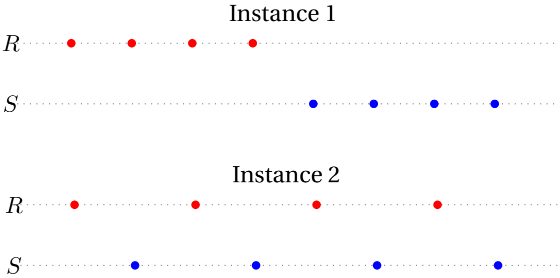
- Input: Two sets R and S of numbers
- Output: $Q := R \cap S$
  - $Q(X) = R(X) \wedge S(X)$
- Worst-case Runtime: $O(\min(|R|, |S|))$
- Some instances are easier than others

Instance 1



Instance-optimal Set Intersection

- Input: Two sets R and S of numbers
- Output: $Q := R \cap S$
  - $Q(X) = R(X) \wedge S(X)$
- Worst-case Runtime: $O(\min(|R|, |S|))$
- Some instances are easier than others



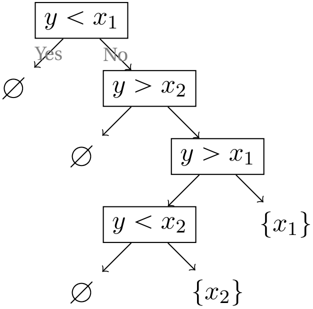
Instance-optimal Set Intersection

- Hwang and Lin, SIAM'72: "Leap-frogging" intersection
- Demaine et al., SODA'00: A form of comparison certificates
- Barbay and Kenyon, SODA'02: "Partition" certificates
- Ngo et al., PODS'14: "Stronger" comparison certificates

Instance-optimal Set Intersection

- Algorithm ⇒ Decision Tree

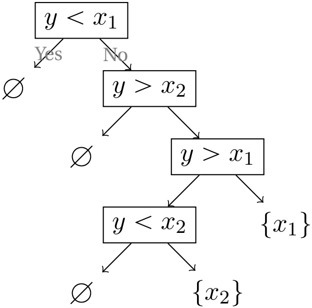
$$\{x_1 < x_2\} \cap \{y\}$$



Instance-optimal Set Intersection

- Algorithm $\Rightarrow$ Decision Tree
- Worst-case runtime $\Rightarrow$ Tree depth

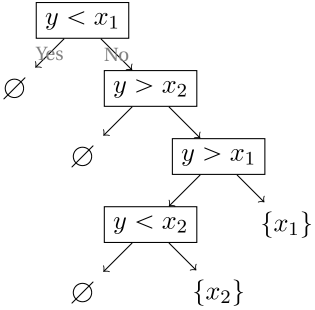
$\{x_1 < x_2\} \cap \{y\}$



Instance-optimal Set Intersection

- Algorithm $\Rightarrow$ Decision Tree
- Worst-case runtime $\Rightarrow$
  Tree depth
- Instance-specific runtime $\Rightarrow$
  Leaf depth

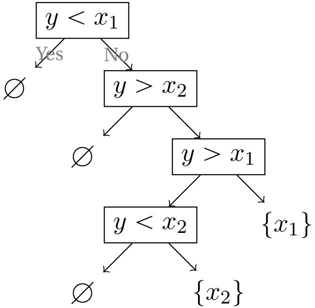
$\{x_1 < x_2\} \cap \{y\}$



Instance-optimal Set Intersection

- Algorithm ⇒ Decision Tree
- Worst-case runtime ⇒
  Tree depth
- Instance-specific runtime ⇒
  Leaf depth
- Instance Certificate ⇒
  Leaf-to-root path

$\{x_1 < x_2\} \cap \{y\}$



Instance-optimal Set Intersection

- Consider the class of algorithms that access the input only through comparisons

Instance-optimal Set Intersection

- Consider the class of algorithms that access the input only through comparisons
  - $R[i] \ \theta \ S[j]$

Instance-optimal Set Intersection

- Consider the class of algorithms that access the input only through comparisons
  - $R[i]\ \theta\ S[j]$
    - $R[i]$ is the i-th smallest element in R

Instance-optimal Set Intersection

- Consider the class of algorithms that access the input only through comparisons
  - $R[i]\ \theta\ S[j]$
    - $R[i]$ is the i-th smallest element in R
    - $S[j]$ is the j-th smallest element in S

Instance-optimal Set Intersection

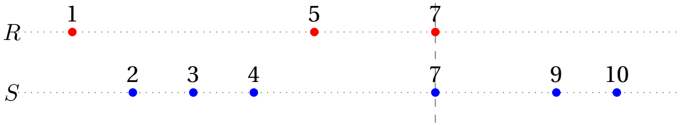
- Consider the class of algorithms that access the input only through comparisons
  - $R[i] \; \theta \; S[j]$
    - $R[i]$ is the i-th smallest element in R
    - $S[j]$ is the j-th smallest element in S
    - $\theta \in \{<, =, >\}$

Comparison-based Certificates

- Input
  - $R = \{1, 5, 7\}$
  - $S = \{2, 3, 4, 7, 9, 10\}$
- Output
  - $Q = \{7\}$

Comparison-based Certificates

- Input
  - $R = \{1, 5, 7\}$
  - $S = \{2, 3, 4, 7, 9, 10\}$
- Output
  - $Q = \{7\}$
- Comparison-based certificate
  - $R[1] < S[1]$
  - $R[2] > S[3]$
  - $R[3] = S[4]$
  - $R[4] = \infty$

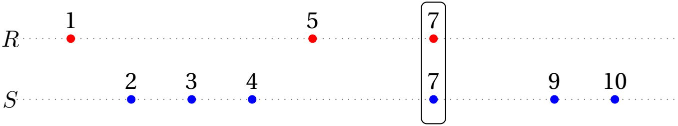


Gap-based Certificates

- $\mathcal{C}_{\Box}$ is a collection of gap intervals from R and S that cover every point *not* in $R \cap S$

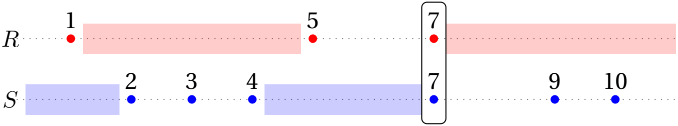
Gap-based Certificates

- $\mathcal{C}_\square$ is a collection of gap intervals from R and S that cover every point *not* in $R \cap S$
  - Input
    - $R = \{1, 5, 7\}$
    - $S = \{2, 3, 4, 7, 9, 10\}$
  - Output
    - $Q = \{7\}$



Gap-based Certificates

- $\mathcal{C}_{\square}$ is a collection of gap intervals from R and S that cover every point *not* in $R \cap S$
  - Input
    - $R = \{1, 5, 7\}$
    - $S = \{2, 3, 4, 7, 9, 10\}$
  - Output
    - $Q = \{7\}$



From $\mathcal{C}_<$ to $\mathcal{C}_\square$

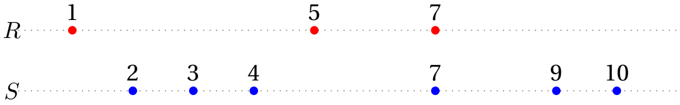
$$|\mathcal{C}_\square| + Z = O(|\mathcal{C}_<|)$$

From $\mathcal{C}_<$ to $\mathcal{C}_\square$

$|\mathcal{C}_\square| + Z = O(|\mathcal{C}_<|)$

Proof idea:

- Take (R, S) and $\mathcal{C}_<$
- $\mathcal{C}_\square \leftarrow \emptyset, \quad \mathcal{Z} \leftarrow \emptyset$



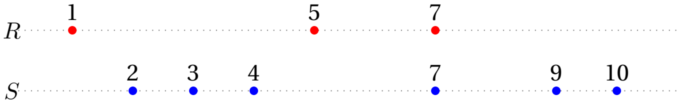
$$\mathcal{C}_< = \{R[1] < S[1], \quad R[2] > S[3], \quad R[3] = S[4], \quad R[4] = \infty\}$$

From $\mathcal{C}_<$ to $\mathcal{C}_\square$

$\boxed{|\mathcal{C}_\square| + Z = O(|\mathcal{C}_<|)}$

Proof idea:

- Take (R, S) and $\mathcal{C}_<$
- $\mathcal{C}_\square \leftarrow \varnothing, \quad \mathcal{Z} \leftarrow \varnothing$
- Repeat: Find t outside $\mathcal{C}_\square \cup \mathcal{Z}$



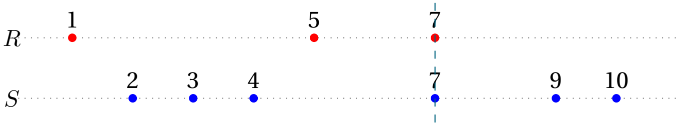
$$\mathcal{C}_< = \{R[1] < S[1], \quad R[2] > S[3], \quad R[3] = S[4], \quad R[4] = \infty\}$$

From $\mathcal{C}_<$ to $\mathcal{C}_\square$

$|\mathcal{C}_\square| + Z = O(|\mathcal{C}_<|)$

Proof idea:

- Take (R, S) and $\mathcal{C}_<$
- $\mathcal{C}_\square \leftarrow \emptyset, \quad \mathcal{Z} \leftarrow \emptyset$
- Repeat: Find t outside $\mathcal{C}_\square \cup \mathcal{Z}$
  - If t is in the output



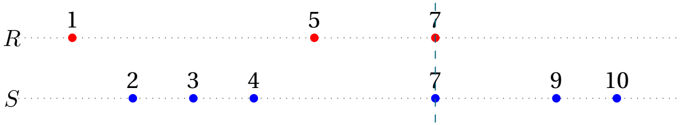
$$\mathcal{C}_< = \{R[1] < S[1], \quad R[2] > S[3], \quad R[3] = S[4], \quad R[4] = \infty\}$$

From $\mathcal{C}_<$ to $\mathcal{C}_\square$

$|\mathcal{C}_\square| + Z = O(|\mathcal{C}_<|)$

Proof idea:

- Take (R, S) and $\mathcal{C}_<$
- $\mathcal{C}_\square \leftarrow \emptyset, \quad \mathcal{Z} \leftarrow \emptyset$
- Repeat: Find t outside $\mathcal{C}_\square \cup \mathcal{Z}$
  - If t is in the output
    - There is $R[i] = S[j] (= t)$



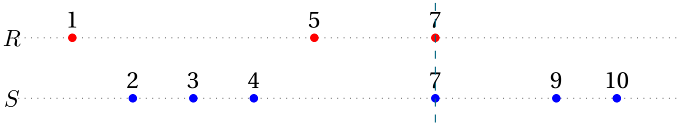
$\mathcal{C}_< = \{R[1] < S[1], \quad R[2] > S[3], \quad R[3] = S[4], \quad R[4] = \infty\}$

From $\mathcal{C}_<$ to $\mathcal{C}_\square$

$|\mathcal{C}_\square| + Z = O(|\mathcal{C}_<|)$

Proof idea:

- Take (R, S) and $\mathcal{C}_<$
- $\mathcal{C}_\square \leftarrow \emptyset, \quad \mathcal{Z} \leftarrow \emptyset$
- Repeat: Find t outside $\mathcal{C}_\square \cup \mathcal{Z}$
  - If t is in the output
    - There is $R[i] = S[j] (= t)$
    - Add t to $\mathcal{Z}$



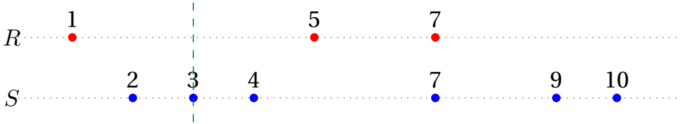
$\mathcal{C}_< = \{R[1] < S[1], \quad R[2] > S[3], \quad R[3] = S[4], \quad R[4] = \infty\}$

From $\mathcal{C}_<$ to $\mathcal{C}_\square$

$|\mathcal{C}_\square| + Z = O(|\mathcal{C}_<|)$

Proof idea:

- Take (R, S) and $\mathcal{C}_<$
- $\mathcal{C}_\square \leftarrow \emptyset, \quad \mathcal{Z} \leftarrow \emptyset$
- Repeat: Find t outside $\mathcal{C}_\square \cup \mathcal{Z}$
  - If t is *not* in the output



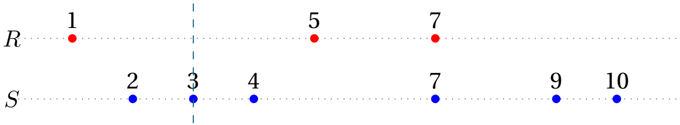
$$\mathcal{C}_< = \{R[1] < S[1], \quad R[2] > S[3], \quad R[3] = S[4], \quad R[4] = \infty\}$$

From $\mathcal{C}_<$ to $\mathcal{C}_\square$

$|\mathcal{C}_\square| + Z = O(|\mathcal{C}_<|)$

Proof idea:

- Take (R, S) and $\mathcal{C}_<$
- $\mathcal{C}_\square \leftarrow \emptyset, \quad \mathcal{Z} \leftarrow \emptyset$
- Repeat: Find t outside $\mathcal{C}_\square \cup \mathcal{Z}$
  - If t is *not* in the output
    - There are *immovable* $R[i] < t < R[i+1]$



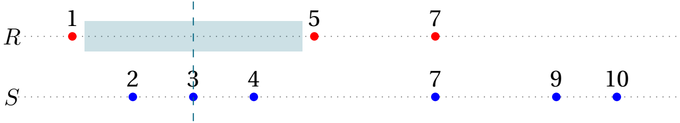
$$\mathcal{C}_< = \{R[1] < S[1], \quad R[2] > S[3], \quad R[3] = S[4], \quad R[4] = \infty\}$$

From $\mathcal{C}_<$ to $\mathcal{C}_\Box$

$|\mathcal{C}_\Box| + Z = O(|\mathcal{C}_<|)$

Proof idea:

- Take (R, S) and $\mathcal{C}_<$
- $\mathcal{C}_\Box \leftarrow \emptyset, \quad \mathcal{Z} \leftarrow \emptyset$
- Repeat: Find t outside $\mathcal{C}_\Box \cup \mathcal{Z}$
  - If t is *not* in the output
    - There are *immovable* $R[i] < t < R[i+1]$
    - Add $(R[i], R[i+1])$ to $\mathcal{C}_\Box$



$\mathcal{C}_< = \{R[1] < S[1], \quad R[2] > S[3], \quad R[3] = S[4], \quad R[4] = \infty\}$

An Instance-Optimal Algorithm for $\cap$

- $\mathcal{C}_{\square} \leftarrow \emptyset$
- $\mathcal{Z} \leftarrow \emptyset$
- Repeat: Find the smallest t outside $\mathcal{C}_{\square} \cup \mathcal{Z}$
  - If t is in the output
    - Add t to $\mathcal{Z}$
  - Otherwise
    - Find $R[i] < t < R[i+1]$
    - Find $S[j] < t < S[j+1]$
    - Add $(R[i], R[i+1])$ and $(S[j], S[j+1])$ to $\mathcal{C}_{\square}$

An Instance-Optimal Algorithm for $\cap$

- $\mathcal{C}_\square \leftarrow \varnothing$
- $\mathcal{Z} \leftarrow \varnothing$
- Repeat: Find the smallest t outside $\mathcal{C}_\square \cup \mathcal{Z}$
  - If t is in the output
    - Add t to $\mathcal{Z}$
  - Otherwise
    - Find $R[i] < t < R[i+1]$
    - Find $S[j] < t < S[j+1]$
    - Add $(R[i], R[i+1])$ and $(S[j], S[j+1])$ to $\mathcal{C}_\square$

Lemma: $|\mathcal{C}_\square| \leqslant 2 \cdot |\mathcal{C}'_\square|, \quad$ for any $\mathcal{C}'_\square$

An Instance-Optimal Algorithm for $\cap$

- $\mathcal{C}_\square \leftarrow \varnothing$
- $\mathcal{Z} \leftarrow \varnothing$
- Repeat: Find the smallest t outside $\mathcal{C}_\square \cup \mathcal{Z}$
  - If t is in the output
    - Add t to $\mathcal{Z}$
  - Otherwise
    - Find $R[i] < t < R[i+1]$
    - Find $S[j] < t < S[j+1]$
    - Add $(R[i], R[i+1])$ and $(S[j], S[j+1])$ to $\mathcal{C}_\square$

Lemma: $|\mathcal{C}_\square| \leqslant 2 \cdot |\mathcal{C}'_\square|, \quad$ for any $\mathcal{C}'_\square$

Runtime: $O(|\mathcal{C}_\square| + Z) = O(|\mathcal{C}_<|)$

Table of Contents

Instance-optimal Database Joins

- Database Join Query

$$Q(\mathbf{X}) = \bigwedge_F R_F(\mathbf{X}_F)$$

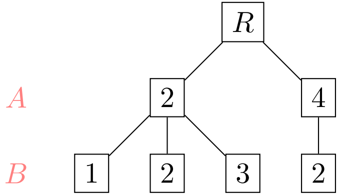
- Examples
  - $Q(A, B) = R(A, B) \wedge S(A) \wedge T(B)$
  - $Q(A, B, C) = R(A, B) \wedge S(B, C) \wedge T(C, A)$
  - $Q(A) = R(A) \wedge S(A)$

Relation Indices $\Rightarrow$ Comparison Certificates

- $R = \{(2,1),(2,2),(2,3),(4,2)\}$
- Suppose $R(A,B)$ is indexed first on A and then on B

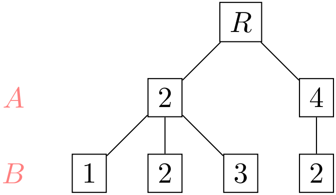
Relation Indices $\Rightarrow$ Comparison Certificates

- $R = \{(2,1),(2,2),(2,3),(4,2)\}$
- Suppose $R(A, B)$ is indexed first on A and then on B



Relation Indices ⇒ Comparison Certificates

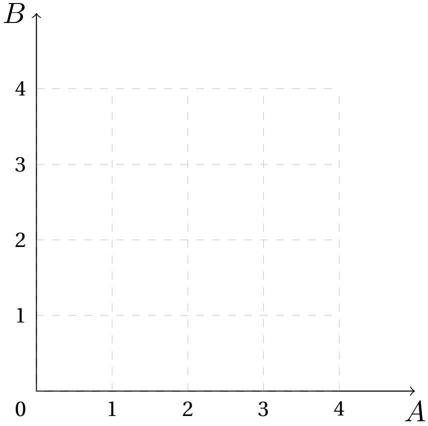
- $R = \{(2,1),(2,2),(2,3),(4,2)\}$
- Suppose $R(A, B)$ is indexed first on A and then on B



$$
\begin{aligned}
R[1] &= 2 \\
R[2] &= 4 \\
R[1,1] &= 1 \\
R[2,1] &= 2 \\
R[2,2] &= \infty
\end{aligned}
$$

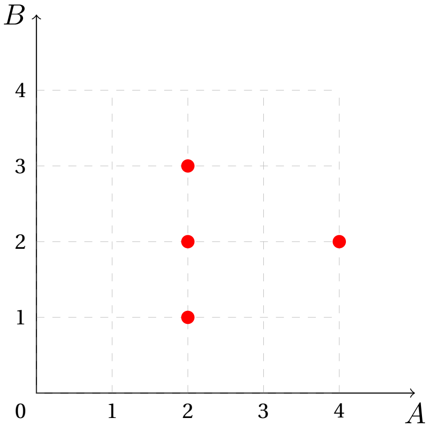
Relation Indices ⇒ Comparison Certificates

- $Q(A, B) = R(A, B) \wedge S(A) \wedge T(B)$



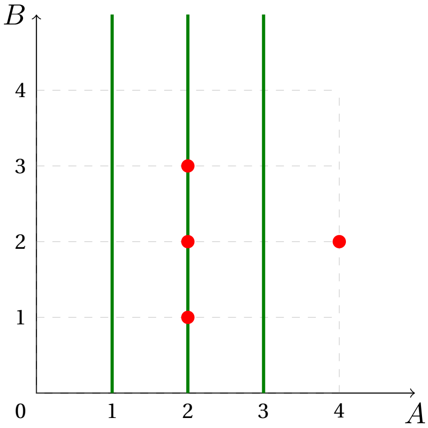
Relation Indices ⇒ Comparison Certificates

- $Q(A, B) = R(A, B) \wedge S(A) \wedge T(B)$
  - $R = \{(2,1), (2,2), (2,3), (4,2)\}$



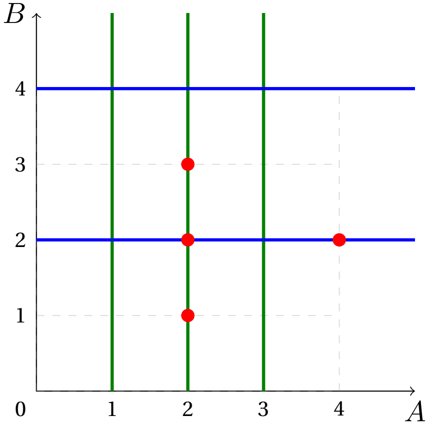
Relation Indices ⇒ Comparison Certificates

- $Q(A, B) = R(A, B) \wedge S(A) \wedge T(B)$
  - $R = \{(2, 1), (2, 2), (2, 3), (4, 2)\}$
  - $S = \{1, 2, 3\}$



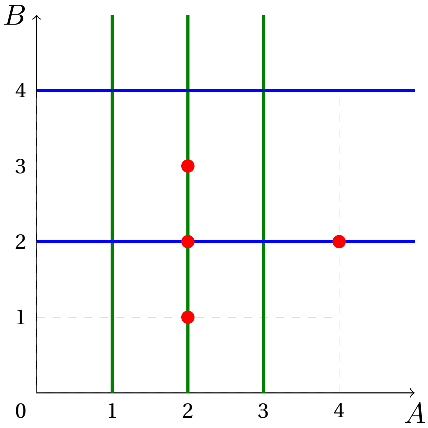
Relation Indices ⇒ Comparison Certificates

- $Q(A, B) = R(A, B) \wedge S(A) \wedge T(B)$
  - $R = \{(2,1), (2,2), (2,3), (4,2)\}$
  - $S = \{1, 2, 3\}$
  - $T = \{2, 4\}$



Relation Indices ⇒ Comparison Certificates

- $Q(A, B) = R(A, B) \wedge S(A) \wedge T(B)$
  - $R = \{(2,1),(2,2),(2,3),(4,2)\}$
  - $S = \{1,2,3\}$
  - $T = \{2,4\}$



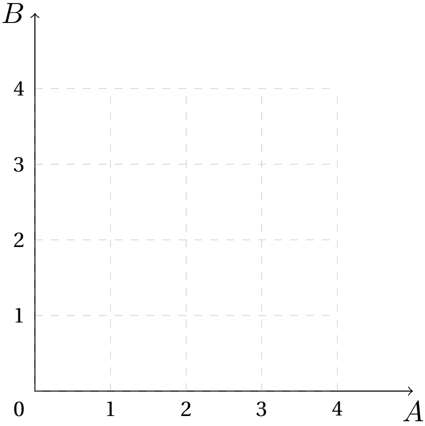
$[A, B]$-Comparison Certificate:

$$\begin{aligned} R[1] &= S[2] \\ R[2] &> S[3] \\ S[4] &= \infty \end{aligned}$$

$$\begin{aligned} T[1] &= R[1,2] \\ T[2] &> R[1,3] \\ R[1,4] &= \infty \end{aligned}$$

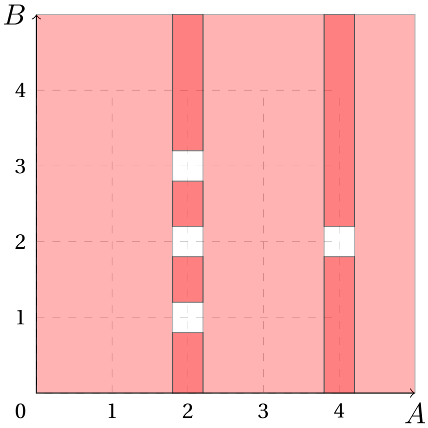
Relation Indices ⇒ Gap Certificates

- $Q(A, B) = R(A, B) \wedge S(A) \wedge T(B)$



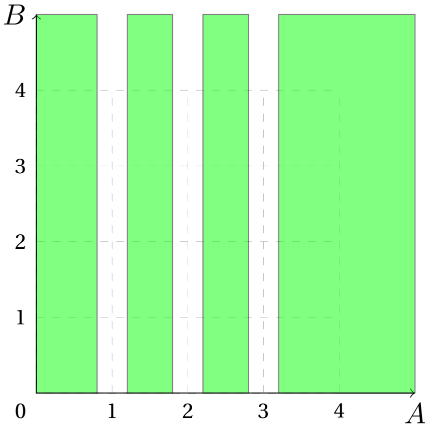
Relation Indices ⇒ Gap Certificates

- $Q(A, B) = R(A, B) \wedge S(A) \wedge T(B)$
  - $R = \{(2, 1), (2, 2), (2, 3), (4, 2)\}$



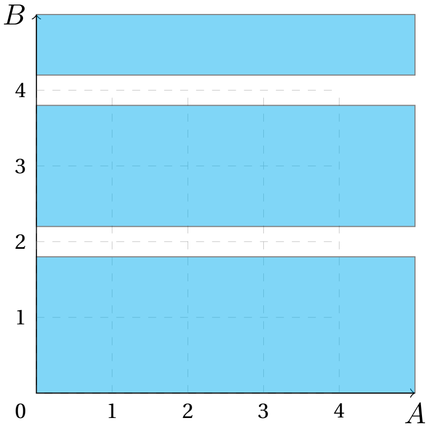
Relation Indices ⇒ Gap Certificates

- $Q(A, B) = R(A, B) \wedge S(A) \wedge T(B)$
  - $R = \{(2,1), (2,2), (2,3), (4,2)\}$
  - $S = \{1, 2, 3\}$



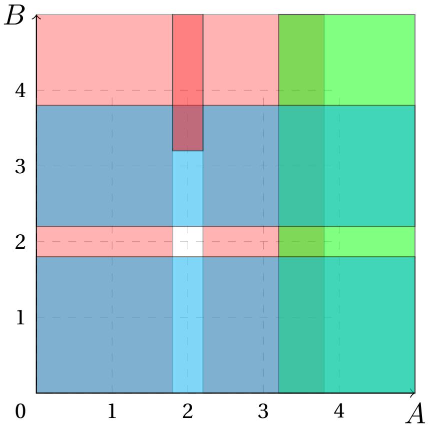
Relation Indices ⇒ Gap Certificates

- $Q(A, B) = R(A, B) \wedge S(A) \wedge T(B)$
  - $R = \{(2,1),(2,2),(2,3),(4,2)\}$
  - $S = \{1,2,3\}$
  - $T = \{2,4\}$



Relation Indices ⇒ Gap Certificates

- $Q(A, B) = R(A, B) \wedge S(A) \wedge T(B)$
  - $R = \{(2, 1), (2, 2), (2, 3), (4, 2)\}$
  - $S = \{1, 2, 3\}$
  - $T = \{2, 4\}$



$[A, B]$-Gap Certificate

Background

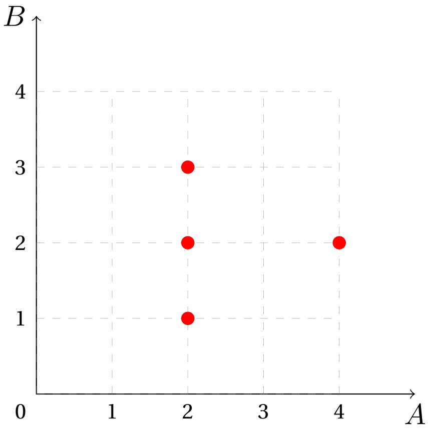
- Ngo et al, PODS'14:
  - $|\mathcal{C}_{\Box}^{\text{gao}}| + Z = O(|\mathcal{C}_{<}^{\text{gao}}|)$
  - Minesweeper algorithm
    - First Instance-optimal Join Algorithm
    - $O(|\mathcal{C}_{<}^{\text{gao}}| + Z)$ for β-acyclic queries
    - $O(|\mathcal{C}_{<}^{\text{gao}}|^{w+1} + Z)$ for treewidth w-queries

Background

- Abo Khamis et al, PODS'15:
  - A tighter notion of certificate $|\mathcal{C}_{\Box}| \leqslant |\mathcal{C}_{\Box}^{\text{gao}}|$
  - Tetris algorithm
    - works over different kinds of indexes.
    - achieves the fractional hypertree-width bound.
    - achieves a series of instance-optimality results.
  - A **proof system** for joins where
    - proof complexity lower bounds/upper bounds are developed.
    - proof sizes precisely capture the runtime of Tetris.

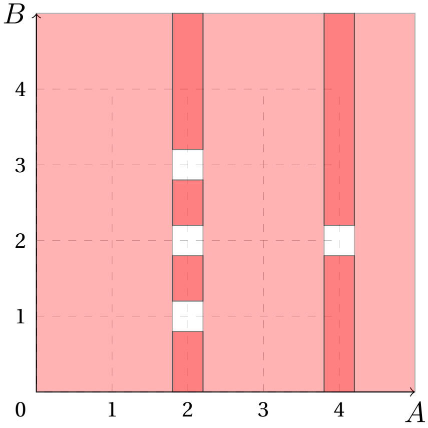
Multiple Indexes $\Rightarrow \mathcal{C}_\square$

$$R = \{(2,1),(2,2),(2,3),(4,2)\}$$



Multiple Indexes $\Rightarrow \mathcal{C}_{\Box}$

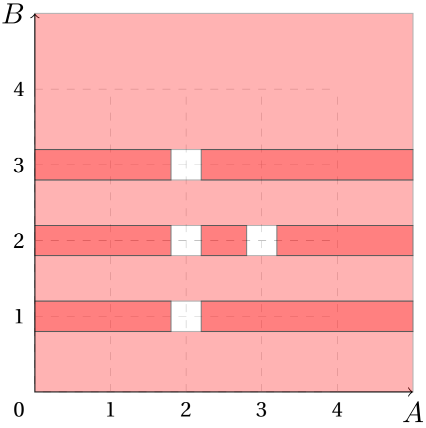
$$R = \{(2,1),(2,2),(2,3),(4,2)\}$$



[A,B]-Index

Multiple Indexes $\Rightarrow \mathcal{C}_{\square}$

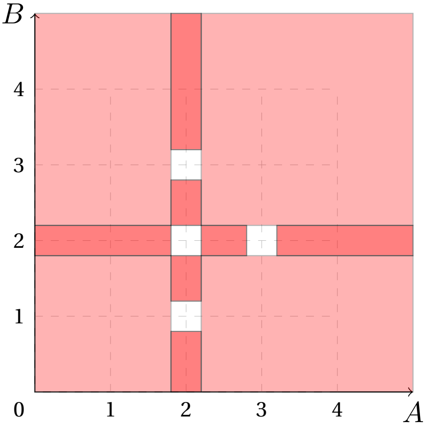
$$R = \{(2,1), (2,2), (2,3), (4,2)\}$$



[B,A]-Index

Multiple Indexes $\Rightarrow \mathcal{C}_\Box$

$$R = \{(2,1),(2,2),(2,3),(4,2)\}$$



Quad tree

Box Cover Problem

Problem (BCP)

Box Cover Problem

Problem (BCP)

Given a set $\mathcal{A}$ of (multi-dimensional rectangular) boxes,

Box Cover Problem

Problem (BCP)

Given a set $\mathcal{A}$ of (multi-dimensional rectangular) boxes,

- list all tuples *not* covered by any box in $\mathcal{A}$.

Box Cover Problem

Problem (BCP)

Given a set $\mathcal{A}$ of (multi-dimensional rectangular) boxes,

- list all tuples *not* covered by any box in $\mathcal{A}$.

Relational Join can be reduced to BCP

BCP Certificates

Definition (Box Certificate)

Given a set of boxes $\mathcal{A}$, a *box certificate* $\mathcal{C}_\square$ for $\mathcal{A}$ is a *minimum-sized* subset of $\mathcal{A}$ such that

$$\bigcup_{\mathbf{c} \in \mathcal{C}_\square} \mathbf{c} = \bigcup_{\mathbf{a} \in \mathcal{A}} \mathbf{a}.$$

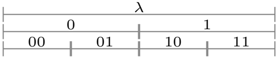
Table of Contents

Dyadic Boxes

- Suppose $|\mathsf{Domain}(A_i)| = 2^d$, for simplicity.

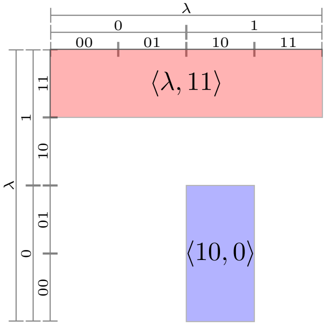
Dyadic Boxes

- Suppose $|\mathsf{Domain}(A_i)| = 2^d$, for simplicity.
- A dyadic interval is a binary string of length $\leqslant d$.



Dyadic Boxes

- Suppose $|\mathsf{Domain}(A_i)| = 2^d$, for simplicity.
- A dyadic interval is a binary string of length $\leqslant d$.
- A dyadic box is an n-tuple of binary strings of length $\leqslant d$.

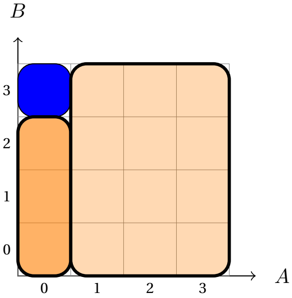


Dyadic Boxes

- Every (not necessarily dyadic) box can be decomposed into $\leqslant (2d)^n = \tilde{O}(1)$ dyadic boxes.

Dyadic Boxes

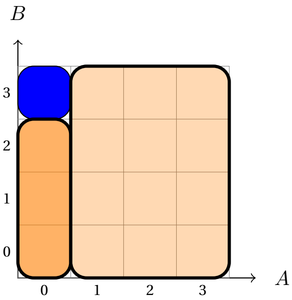
- Every (not necessarily dyadic) box can be decomposed into $\leqslant (2d)^n = \tilde{O}(1)$ dyadic boxes.



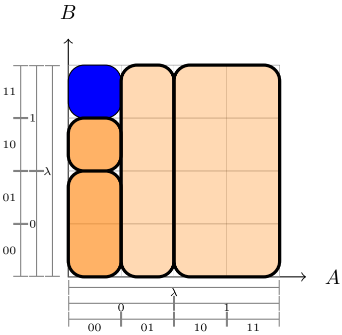
Gap boxes for $R(A, B)$

Dyadic Boxes

- Every (not necessarily dyadic) box can be decomposed into $\leqslant (2d)^n = \tilde{O}(1)$ dyadic boxes.



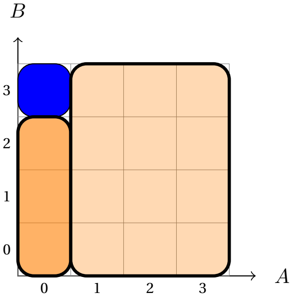
Gap boxes for $R(A, B)$



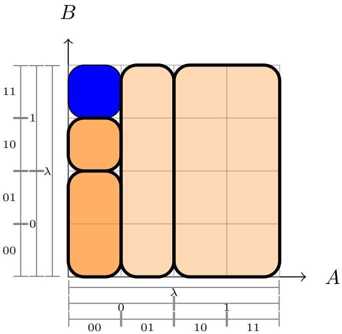
Corresponding dyadic boxes

Dyadic Boxes

- Every (not necessarily dyadic) box can be decomposed into $\leqslant (2d)^n = \tilde{O}(1)$ dyadic boxes.



Gap boxes for $R(A, B)$



Corresponding dyadic boxes

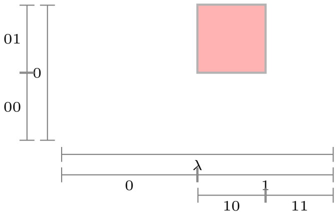
- Every n-tuple is contained in $\leqslant d^n = \tilde{O}(1)$ dyadic boxes.

Geometric Resolution ...

... is an inference system for BCP.

Geometric Resolution ...

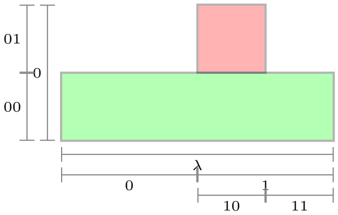
... is an inference system for BCP.



$\langle 10, 01 \rangle$

Geometric Resolution ...

... is an inference system for BCP.

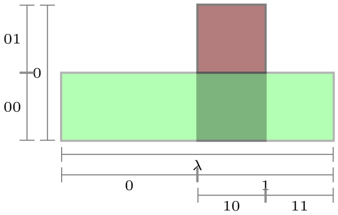


$\langle 10, 01 \rangle$

$\langle \lambda, 00 \rangle$

Geometric Resolution …

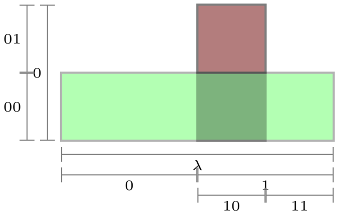
… is an inference system for BCP.



$$\frac{\color{red}{\langle 10, 01\rangle} \quad \color{green}{\langle \lambda, 00\rangle}}{\langle 10, 0\rangle}$$

Geometric Resolution ...

... is an inference system for BCP.

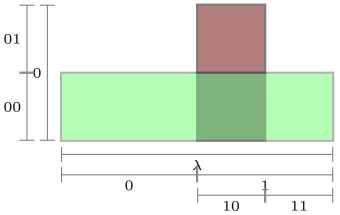


$$\frac{\begin{array}{c}\langle 10, 01\rangle \\ \langle \lambda, 00\rangle\end{array}}{\langle 10, 0\rangle}$$

... is analogous to traditional resolution in logic.

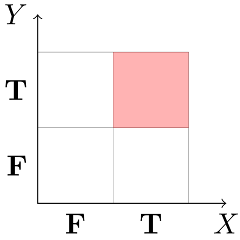
Geometric Resolution ...

... is an inference system for BCP.



$$\frac{\color{red}{\langle 10, 01\rangle} \quad \color{green}{\langle \lambda, 00\rangle}}{\langle 10, 0\rangle}$$

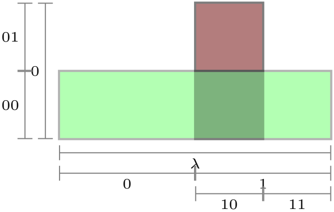
... is analogous to traditional resolution in logic.



$\color{red}{\bar{X} \vee \bar{Y}}$

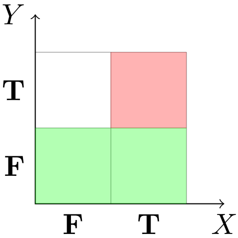
Geometric Resolution ...

... is an inference system for BCP.



$$\frac{\langle 10, 01\rangle \quad \langle \lambda, 00\rangle}{\langle 10, 0\rangle}$$

... is analogous to traditional resolution in logic.

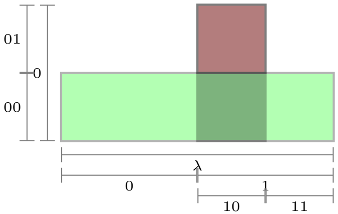


$\bar{X} \vee \bar{Y}$

Y

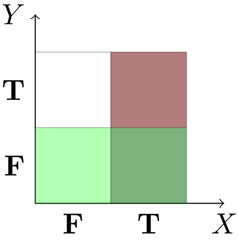
Geometric Resolution ...

... is an inference system for BCP.



$$\frac{\begin{array}{c}\langle 10, 01\rangle \\ \langle \lambda, 00\rangle\end{array}}{\langle 10, 0\rangle}$$

... is analogous to traditional resolution in logic.



$$\frac{\begin{array}{c}\bar{X} \vee \bar{Y} \\ Y\end{array}}{\bar{X}}$$

Geometric Resolution

- Geometric Resolution is complete.

Geometric Resolution

- Geometric Resolution is complete.
  - Given a set of boxes $\mathcal{A}$ that covers some box $\mathbf{b}$, we can infer from $\mathcal{A}$ a box $\mathbf{b}'$ that covers $\mathbf{b}$.

Geometric Resolution

- Geometric Resolution is complete.
  - Given a set of boxes $\mathcal{A}$ that covers some box $\mathbf{b}$, we can infer from $\mathcal{A}$ a box $\mathbf{b}'$ that covers $\mathbf{b}$.
- Three main variations:

Geometric Resolution

- Geometric Resolution is complete.
  - Given a set of boxes $\mathcal{A}$ that covers some box b, we can infer from $\mathcal{A}$ a box b' that covers b.
- Three main variations:
  - GEOMETRIC RESOLUTION

Geometric Resolution

- Geometric Resolution is complete.
  - Given a set of boxes $\mathcal{A}$ that covers some box b, we can infer from $\mathcal{A}$ a box b' that covers b.
- Three main variations:
  - GEOMETRIC RESOLUTION
  - ORDERED GEOMETRIC RESOLUTION

Geometric Resolution

- Geometric Resolution is complete.
  - Given a set of boxes $\mathcal{A}$ that covers some box b, we can infer from $\mathcal{A}$ a box b′ that covers b.
- Three main variations:
  - Geometric Resolution
  - Ordered Geometric Resolution
  - Tree Ordered Geometric Resolution

(General) Geometric Resolution

$$\mathbf{w} = \mathsf{Resolve}(\mathbf{w}_1, \mathbf{w}_2)$$

$$\begin{array}{rcl}
\mathbf{w}_1 & = & \langle y_1 \;,\; \ldots \;,\; y_{\ell-1} \;,\; x_\ell 0 \;,\; y_{\ell+1} \;,\; \ldots \;,\; y_n \rangle \\
\mathbf{w}_2 & = & \langle z_1 \;,\; \ldots \;,\; z_{\ell-1} \;,\; x_\ell 1 \;,\; z_{\ell+1} \;,\; \ldots \;,\; z_n \rangle \\
\hline
\mathbf{w} & = & \langle \ldots \;,\; y_{\ell-1} \cap z_{\ell-1} \;,\; x_\ell \;,\; y_{\ell+1} \cap z_{\ell+1} \;,\; \ldots \rangle
\end{array}$$

Ordered Geometric Resolution

$$\mathbf{w} = \mathsf{Resolve}(\mathbf{w}_1, \mathbf{w}_2)$$

$$\begin{array}{rcl}
\mathbf{w}_1 & = & \langle y_1 \ , \ \ldots \ , \ y_{\ell-1} \ , \ x_\ell 0 \ , \ \lambda \ , \ \ldots \ , \ \lambda\rangle \\
\mathbf{w}_2 & = & \langle z_1 \ , \ \ldots \ , \ z_{\ell-1} \ , \ x_\ell 1 \ , \ \lambda \ , \ \ldots \ , \ \lambda\rangle \\
\hline
\mathbf{w} & = & \langle \ldots \ , \ y_{\ell-1} \cap z_{\ell-1} \ , \ x_\ell \ , \ \lambda \ , \ \ldots \ , \ \lambda\rangle
\end{array}$$

Tree-Ordered Geometric Resolution

- Proof is a Tree (as opposed to DAG)
  - No caching

Table of Contents

Tetris: a recursive algorithm for BCP

Tetris: a recursive algorithm for BCP

is b covered by the union of boxes in $\mathcal{A}$?

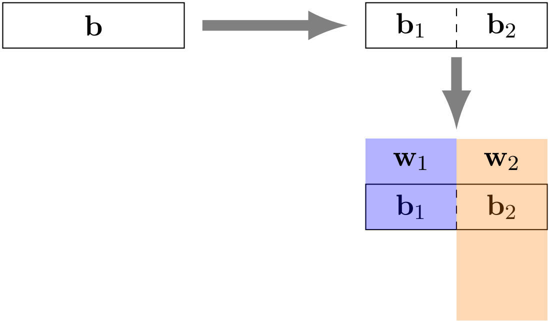
$\boxed{\mathrm{b}}$

Tetris: a recursive algorithm for BCP

split $\mathbf{b}$ into two halves $\mathbf{b}_1, \mathbf{b}_2$,

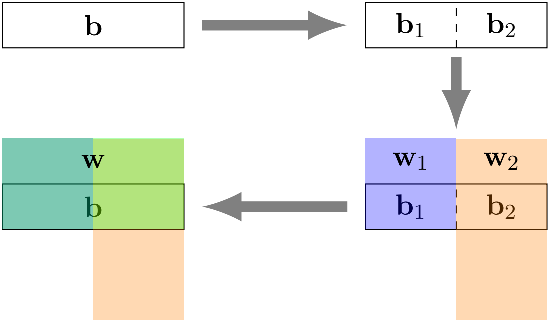
Tetris: a recursive algorithm for BCP

recursively verify that $\mathbf{b}_1$ and $\mathbf{b}_2$ are covered
through finding two witnesses $\mathbf{w}_1, \mathbf{w}_2$ that cover $\mathbf{b}_1, \mathbf{b}_2$,



Tetris: a recursive algorithm for BCP

$\mathbf{w} = \mathsf{Resolve}(\mathbf{w}_1, \mathbf{w}_2)$, then $\mathbf{w}$ covers $\mathbf{b}$,
add $\mathbf{w}$ to $\mathcal{A}$, $\mathbf{w}$ is a witness for $\mathbf{b}$.



Two main analytical components

- runtime $= \Theta(\#\text{resolutions})$

Two main analytical components

- runtime $= \Theta(\#\text{resolutions})$
- #resolutions is a function of dimension ordering

Two main analytical components

- runtime $= \Theta(\#\text{resolutions})$
- #resolutions is a function of dimension ordering
- Different initializations lead to different results

Two main analytical components

- runtime $= \Theta(\#\text{resolutions})$
- #resolutions is a function of dimension ordering
- Different initializations lead to different results
  - Tetris-Preloaded (load all input boxes)

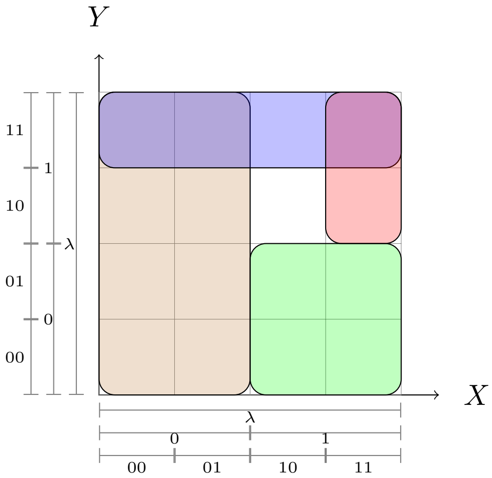
Two main analytical components

- runtime $= \Theta(\#\text{resolutions})$
- #resolutions is a function of dimension ordering
- Different initializations lead to different results
  - Tetris-Preloaded (load all input boxes)
  - Tetris-Reloaded (load as needed)

Two main analytical components

- runtime $= \Theta(\#\text{resolutions})$
- #resolutions is a function of dimension ordering
- Different initializations lead to different results
  - Tetris-Preloaded (load all input boxes)
  - Tetris-Reloaded (load as needed)
  - Tetris-Balanced (work under a transformed space)

Tetris-Preloaded: Example



Input Boxes

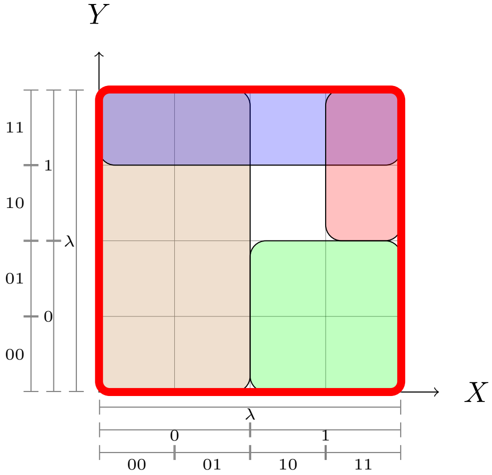
$\langle 0, \lambda \rangle$

$\langle 1, 0 \rangle$

$\langle \lambda, 11 \rangle$

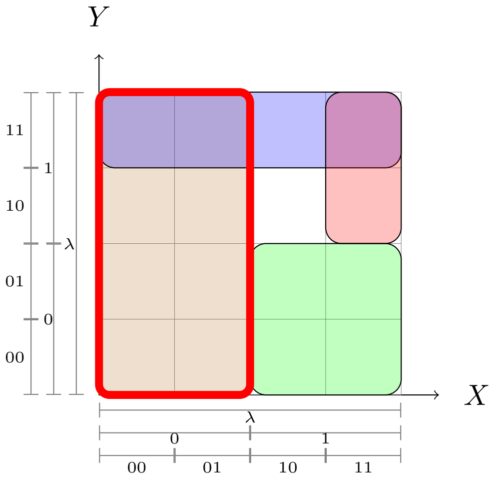
$\langle 11, 1 \rangle$

Tetris-Preloaded: Example



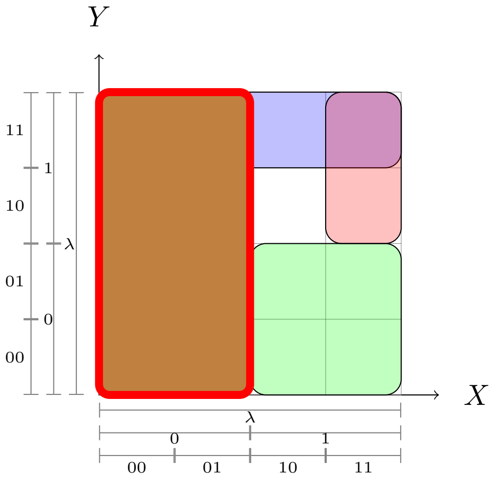
Is $\langle \lambda, \lambda \rangle$ covered?
No
Split into $\langle 0, \lambda \rangle$ and $\langle 1, \lambda \rangle$

Tetris-Preloaded: Example



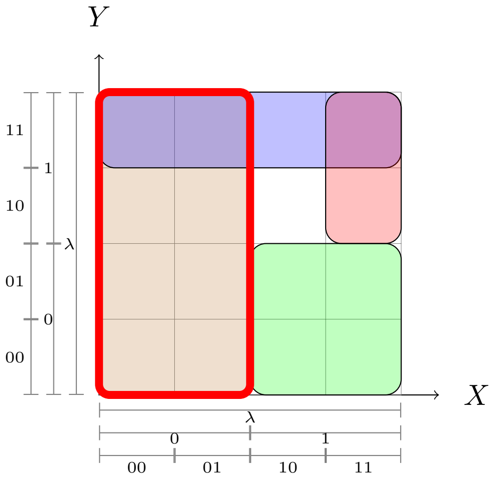
Is $\langle 0, \lambda \rangle$ covered?

Tetris-Preloaded: Example



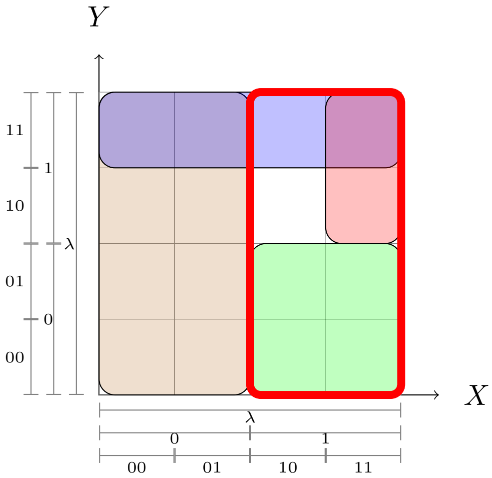
Is $\langle 0, \lambda \rangle$ covered?
Yes by $\langle 0, \lambda \rangle$

Tetris-Preloaded: Example



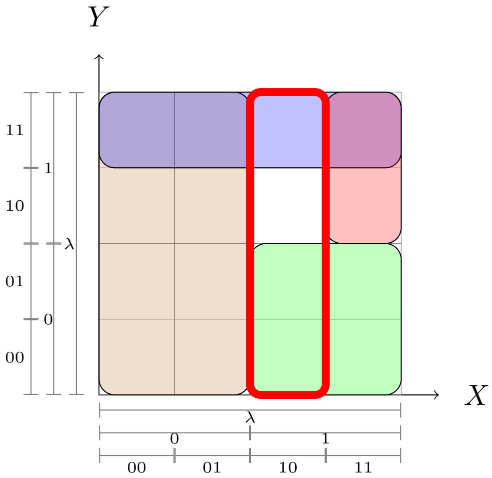
Is $\langle 0, \lambda \rangle$ covered?
Yes by $\langle 0, \lambda \rangle$

Tetris-Preloaded: Example



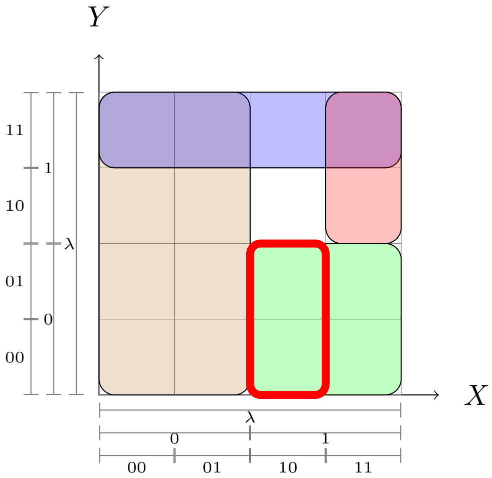
Is $\langle 1, \lambda \rangle$ covered?
No
Split into $\langle 10, \lambda \rangle$ and $\langle 11, \lambda \rangle$

Tetris-Preloaded: Example



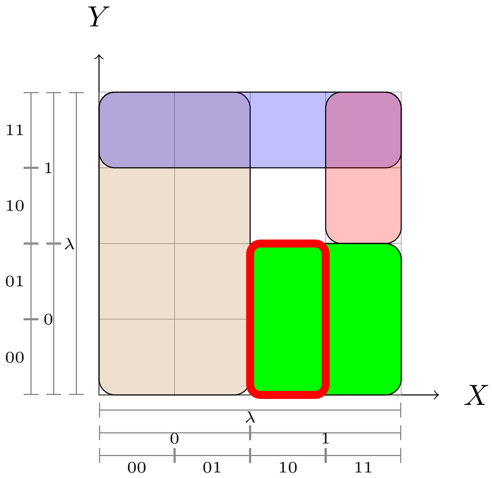
Is $\langle 10, \lambda \rangle$ covered?
No
Split into $\langle 10, 0 \rangle$ and $\langle 10, 1 \rangle$

Tetris-Preloaded: Example



Is $\langle 10, 0\rangle$ covered?

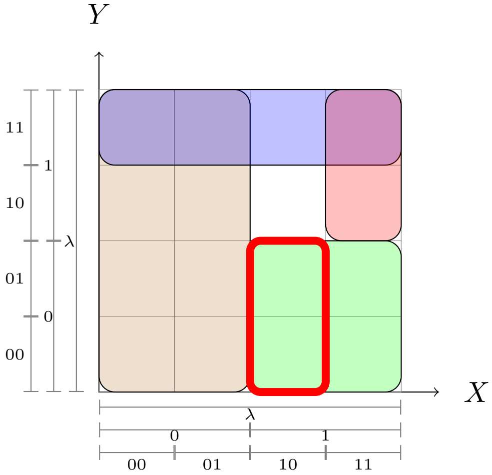
Tetris-Preloaded: Example



Is $\langle 10, 0\rangle$ covered?

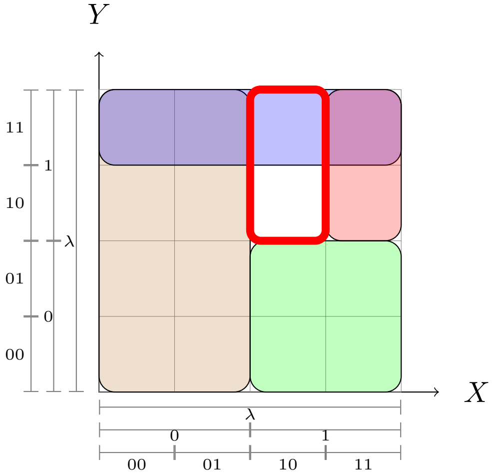
Yes by $\langle 1, 0\rangle$

Tetris-Preloaded: Example



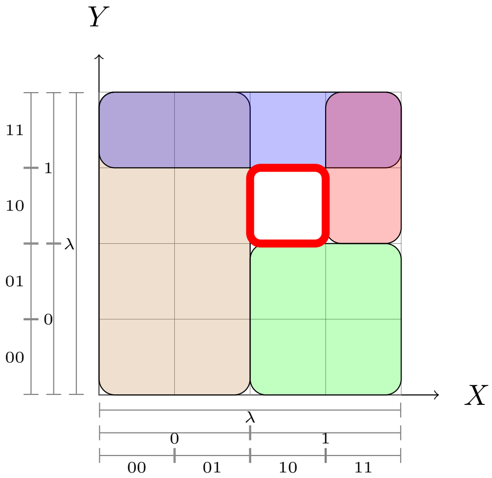
Is $\langle 10, 0\rangle$ covered?
Yes by $\langle 1, 0\rangle$

Tetris-Preloaded: Example



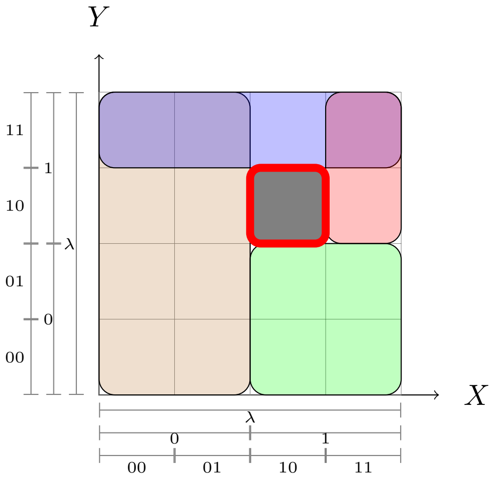
Is $\langle 10, 1\rangle$ covered?
No
Split into $\langle 10, 10\rangle$ and $\langle 10, 11\rangle$

Tetris-Preloaded: Example



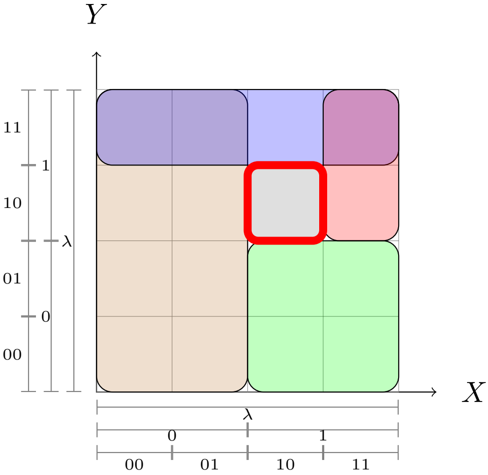
Is $\langle 10, 10 \rangle$ covered?
No
It cannot be split

Tetris-Preloaded: Example



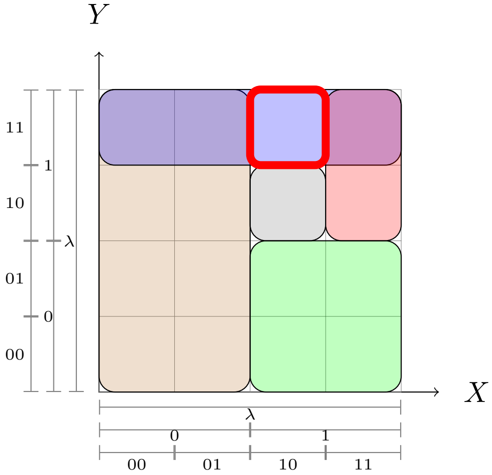
Is $\langle 10, 10 \rangle$ covered?
No
It cannot be split
Output $\langle 10, 10 \rangle$
Add a box $\langle 10, 10 \rangle$

Tetris-Preloaded: Example



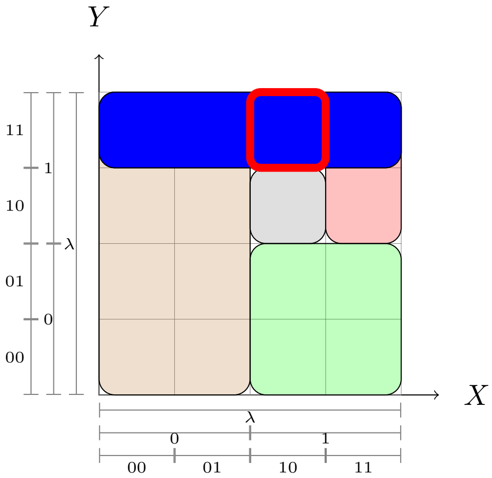
Is $\langle 10, 10\rangle$ covered?
No
It cannot be split
Output $\langle 10, 10\rangle$
Add a box $\langle 10, 10\rangle$

Tetris-Preloaded: Example



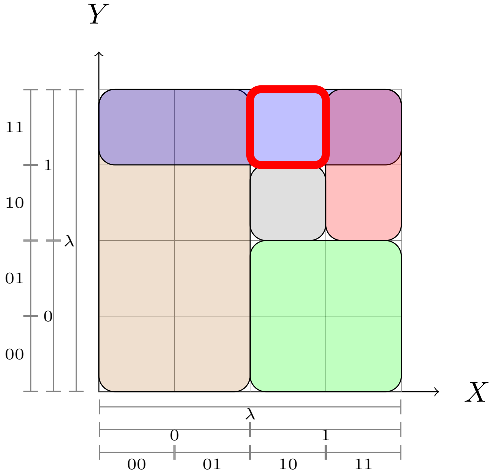
Is $\langle 10, 11 \rangle$ covered?

Tetris-Preloaded: Example



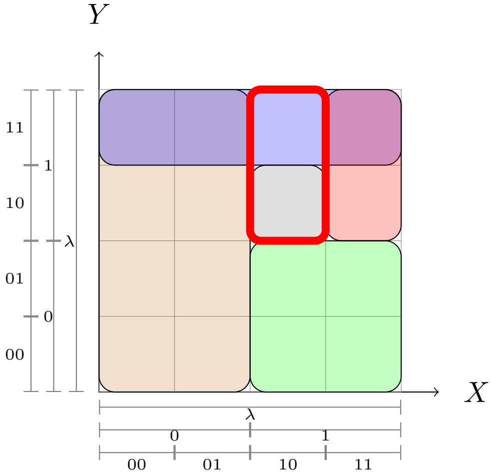
Is $\langle 10, 11 \rangle$ covered?
Yes by $\langle \lambda, 11 \rangle$

Tetris-Preloaded: Example



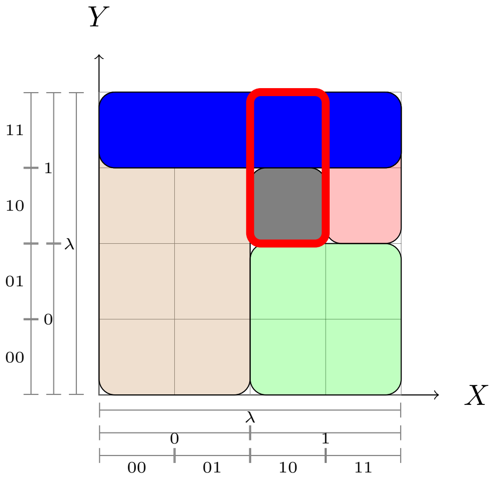
Is $\langle 10, 11 \rangle$ covered?
Yes by $\langle \lambda, 11 \rangle$

Tetris-Preloaded: Example



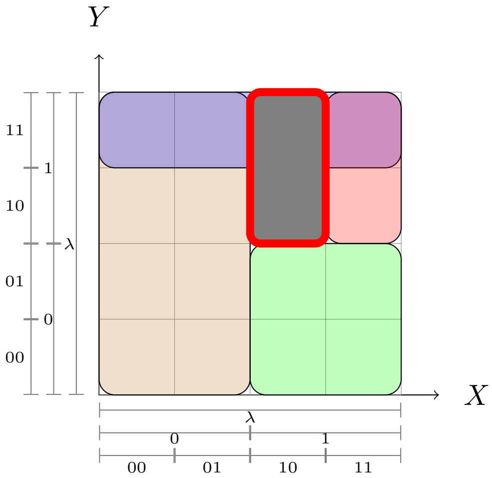
Backtrack to $\langle 10, 1 \rangle$

Tetris-Preloaded: Example



Backtrack to $\langle 10, 1 \rangle$
Resolve
$\langle \lambda, 11 \rangle$
$\langle 10, 10 \rangle$

Tetris-Preloaded: Example



Backtrack to $\langle 10, 1\rangle$

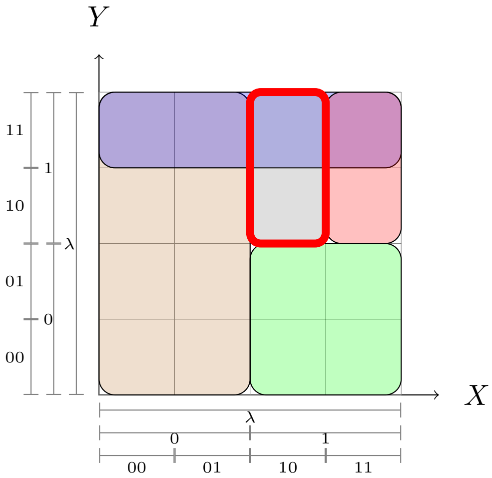
Resolve

$\langle \lambda, 11\rangle$

$\langle 10, 10\rangle$

$\langle 10, 1\rangle$

Tetris-Preloaded: Example



Backtrack to $\langle 10, 1 \rangle$

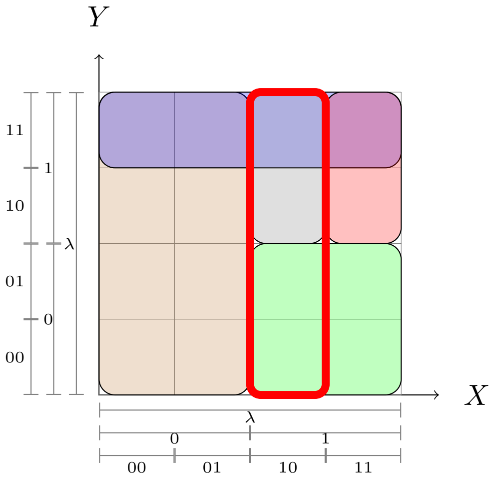
Resolve

$\langle \lambda, 11 \rangle$

$\langle 10, 10 \rangle$

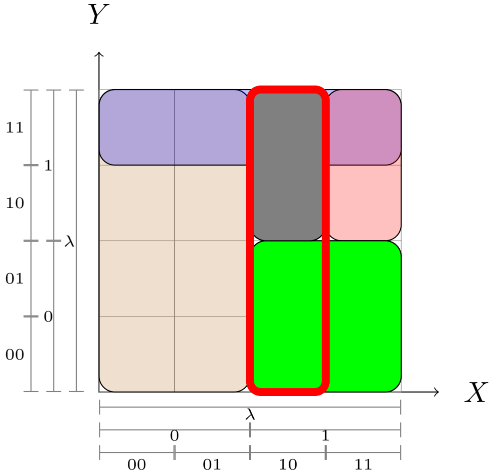
$\langle 10, 1 \rangle$

Tetris-Preloaded: Example



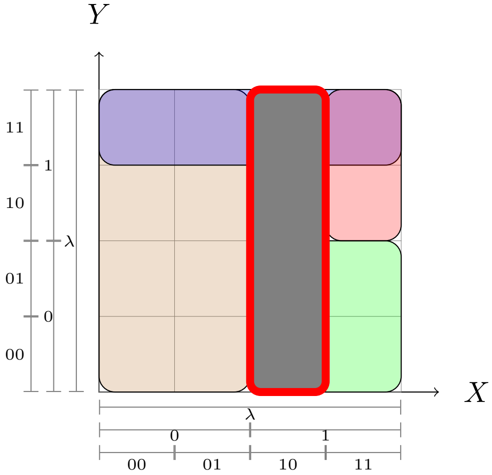
Backtrack to $\langle 10, \lambda \rangle$

Tetris-Preloaded: Example



Backtrack to $\langle 10, \lambda \rangle$
Resolve
$\langle 10, 1 \rangle$
$\underline{\langle 1, 0 \rangle}$

Tetris-Preloaded: Example



Backtrack to $\langle 10, \lambda \rangle$

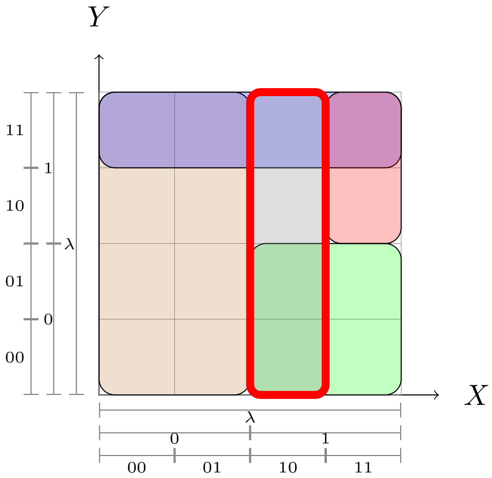
Resolve

$\langle 10, 1 \rangle$

$\langle 1, 0 \rangle$

$\overline{\langle 10, \lambda \rangle}$

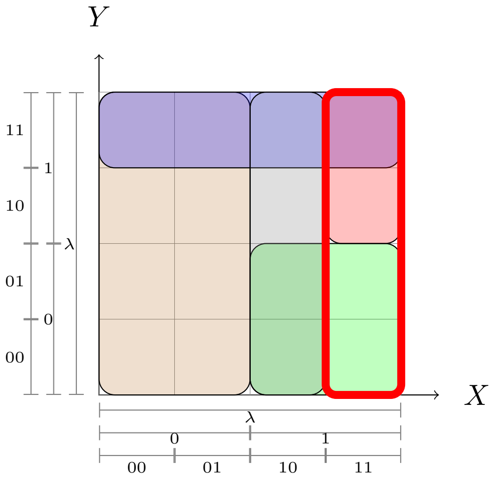
Tetris-Preloaded: Example



Backtrack to $\langle 10, \lambda \rangle$
Resolve

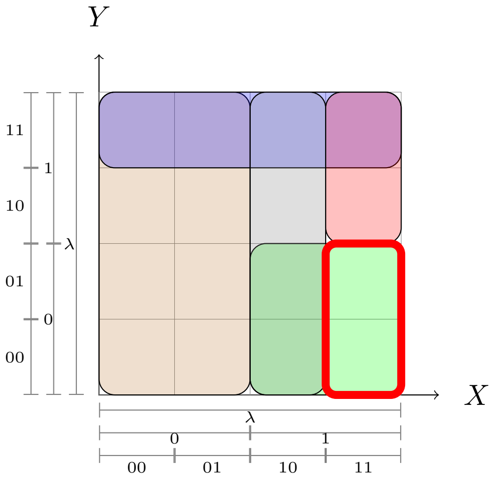
$$\frac{\begin{array}{l}\langle 10, 1 \rangle \\ \langle 1, 0 \rangle\end{array}}{\langle 10, \lambda \rangle}$$

Tetris-Preloaded: Example



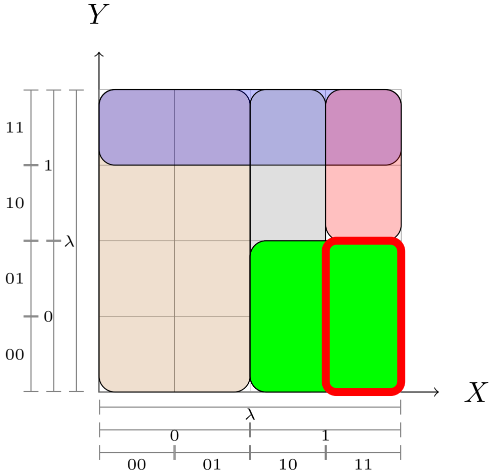
Is $\langle 11, \lambda \rangle$ covered?
No
Split into $\langle 11, 0 \rangle$ and $\langle 11, 1 \rangle$

Tetris-Preloaded: Example



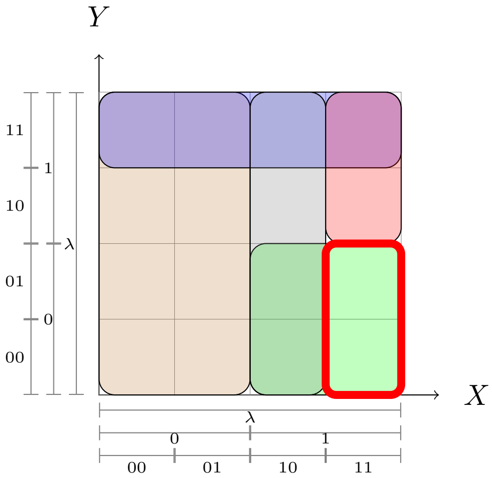
Is $\langle 11, 0\rangle$ covered?

Tetris-Preloaded: Example



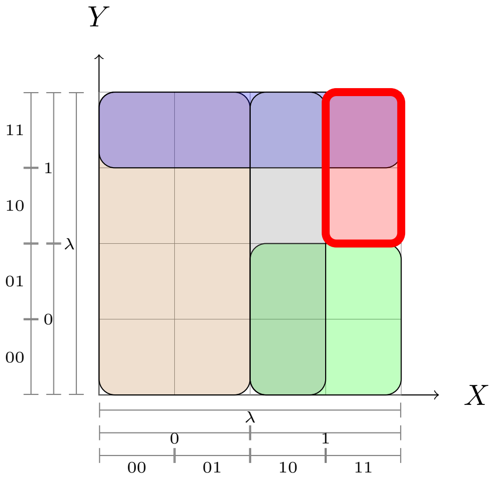
Is $\langle 11, 0\rangle$ covered?
Yes by $\langle 1, 0\rangle$

Tetris-Preloaded: Example



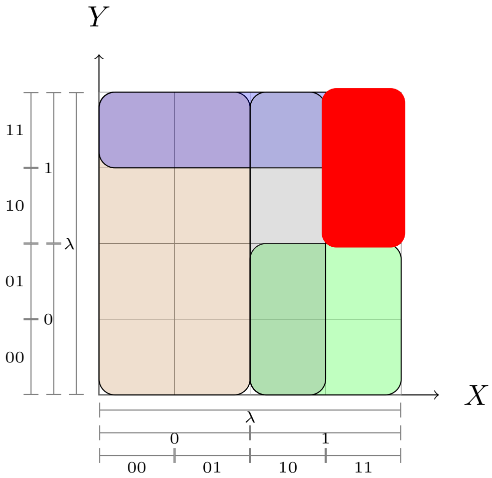
Is $\langle 11, 0\rangle$ covered?
Yes by $\langle 1, 0\rangle$

Tetris-Preloaded: Example



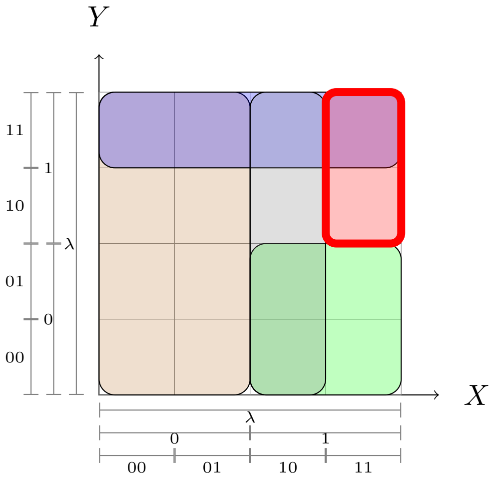
Is $\langle 11, 1 \rangle$ covered?

Tetris-Preloaded: Example



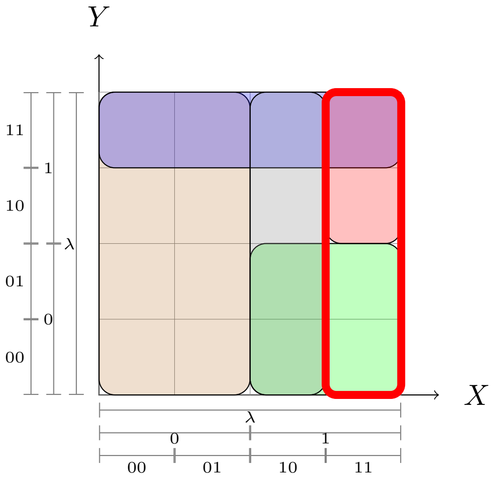
Is $\langle 11, 1 \rangle$ covered?
Yes by $\langle 11, 1 \rangle$

Tetris-Preloaded: Example



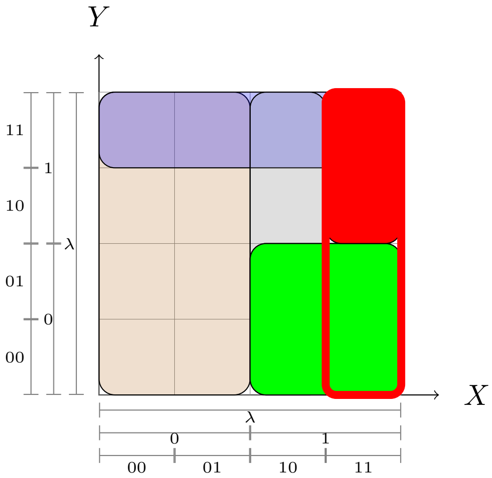
Is $\langle 11, 1 \rangle$ covered?
Yes by $\langle 11, 1 \rangle$

Tetris-Preloaded: Example



Backtrack to $\langle 11, \lambda \rangle$

Tetris-Preloaded: Example



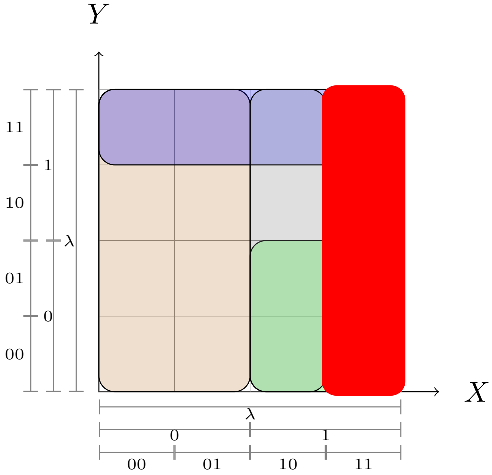
Backtrack to $\langle 11, \lambda \rangle$

Resolve

$\langle 11, 1 \rangle$

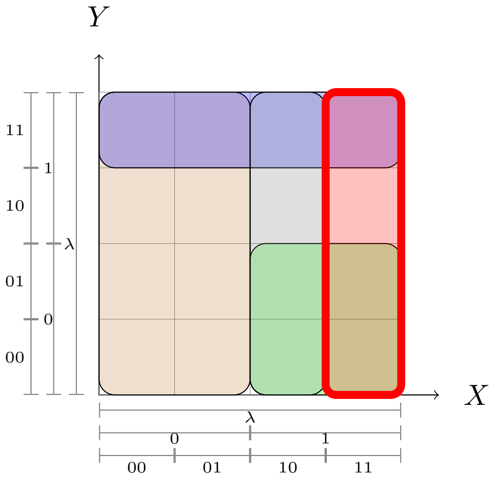
$\langle 1, 0 \rangle$

Tetris-Preloaded: Example



Backtrack to $\langle 11, \lambda \rangle$
Resolve
$\langle 11, 1 \rangle$
$\langle 1, 0 \rangle$
$\overline{\langle 11, \lambda \rangle}$

Tetris-Preloaded: Example

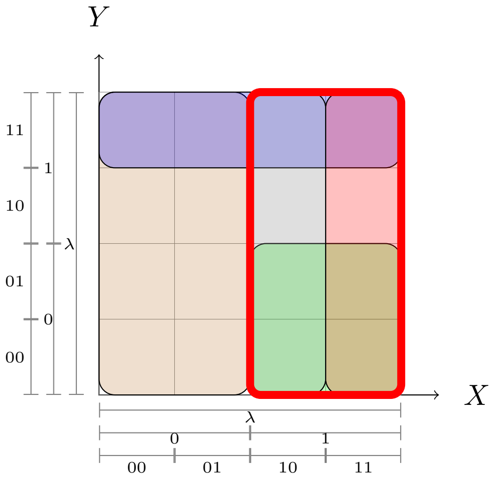


Backtrack to $\langle 11, \lambda \rangle$

Resolve

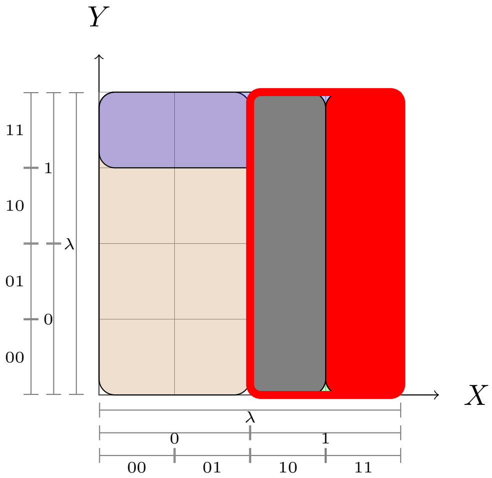
$$\frac{\langle 11, 1 \rangle \quad \langle 1, 0 \rangle}{\langle 11, \lambda \rangle}$$

Tetris-Preloaded: Example



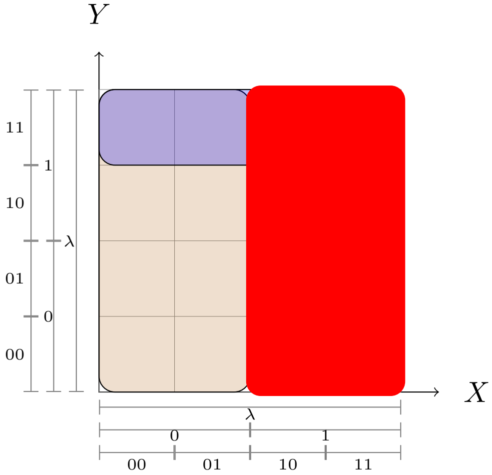
Backtrack to $\langle 1, \lambda \rangle$

Tetris-Preloaded: Example



Backtrack to $\langle 1, \lambda \rangle$
Resolve
$\langle 10, \lambda \rangle$
$\langle 11, \lambda \rangle$

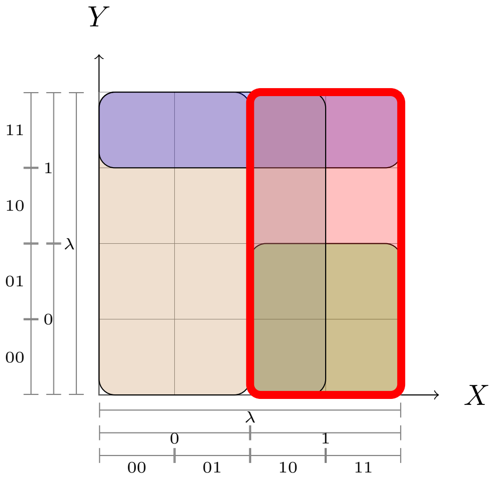
Tetris-Preloaded: Example



Backtrack to $\langle 1, \lambda \rangle$
Resolve
$\langle 10, \lambda \rangle$
$\langle 11, \lambda \rangle$

$\langle 1, \lambda \rangle$

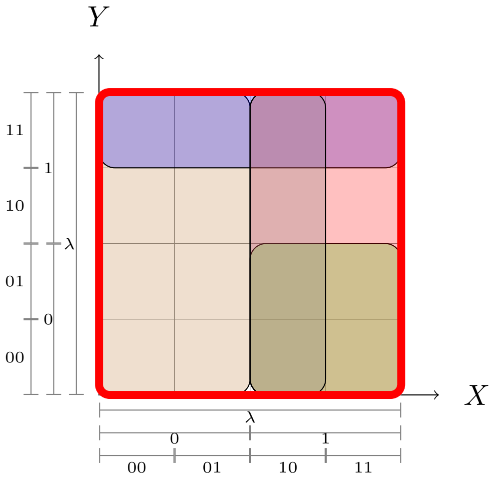
Tetris-Preloaded: Example



Backtrack to $\langle 1, \lambda \rangle$
Resolve

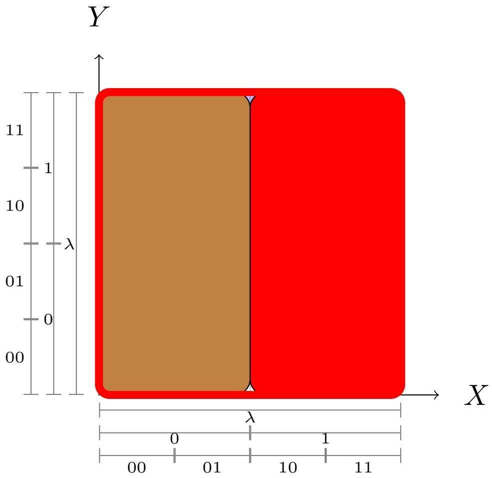
$$\frac{\langle 10, \lambda \rangle \quad \langle 11, \lambda \rangle}{\langle 1, \lambda \rangle}$$

Tetris-Preloaded: Example



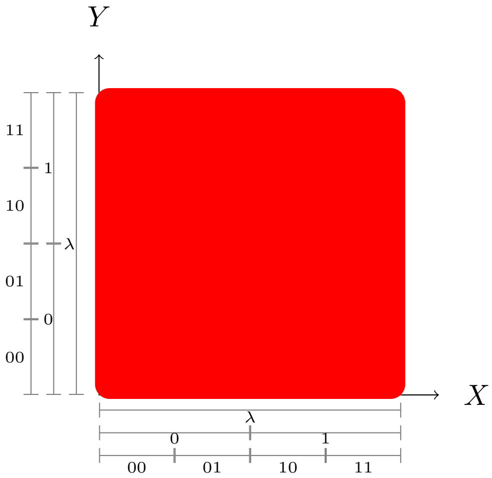
Backtrack to $\langle \lambda, \lambda \rangle$

Tetris-Preloaded: Example



Backtrack to $\langle \lambda, \lambda \rangle$
Resolve
$\langle 0, \lambda \rangle$
$\langle 1, \lambda \rangle$

Tetris-Preloaded: Example



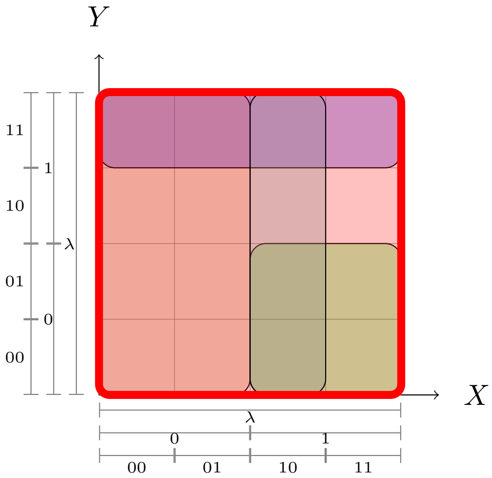
Backtrack to $\langle \lambda, \lambda \rangle$
Resolve

$\langle 0, \lambda \rangle$

$\langle 1, \lambda \rangle$

$\langle \lambda, \lambda \rangle$

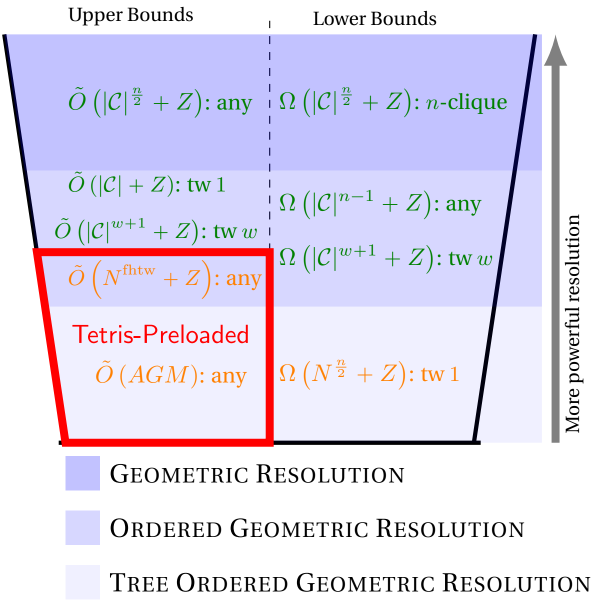
Tetris-Preloaded: Example



Backtrack to $\langle \lambda, \lambda \rangle$
Resolve

$$\frac{\begin{array}{c}\langle 0, \lambda \rangle \\ \langle 1, \lambda \rangle\end{array}}{\langle \lambda, \lambda \rangle}$$

Done!

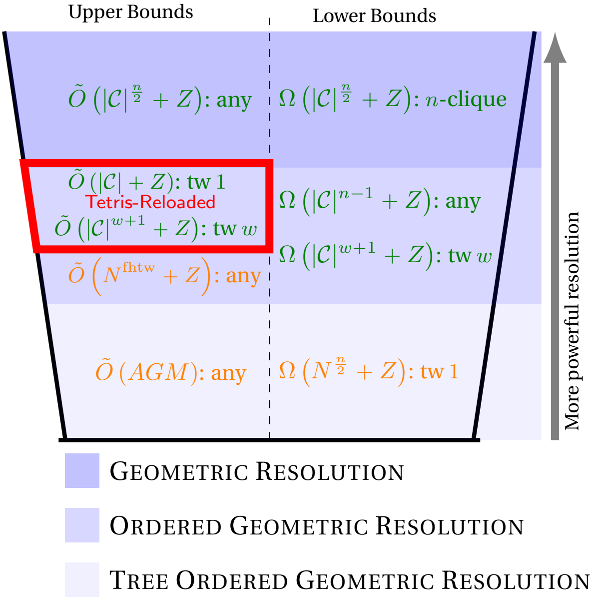

Upper Bounds
Lower Bounds
$\tilde{O}\left(|\mathcal{C}|^{\frac{n}{2}} + Z\right)$: any
$\Omega\left(|\mathcal{C}|^{\frac{n}{2}} + Z\right)$: n-clique
$\tilde{O}\left(|\mathcal{C}| + Z\right)$: tw 1
$\Omega\left(|\mathcal{C}|^{n-1} + Z\right)$: any
$\tilde{O}\left(|\mathcal{C}|^{w+1} + Z\right)$: tw w
$\Omega\left(|\mathcal{C}|^{w+1} + Z\right)$: tw w
$\tilde{O}\left(N^{\text{fhtw}} + Z\right)$: any
Tetris-Preloaded
$\tilde{O}\left(AGM\right)$: any
$\Omega\left(N^{\frac{n}{2}} + Z\right)$: tw 1
More powerful resolution
Geometric Resolution
Ordered Geometric Resolution
Tree Ordered Geometric Resolution

Tetris-Reloaded

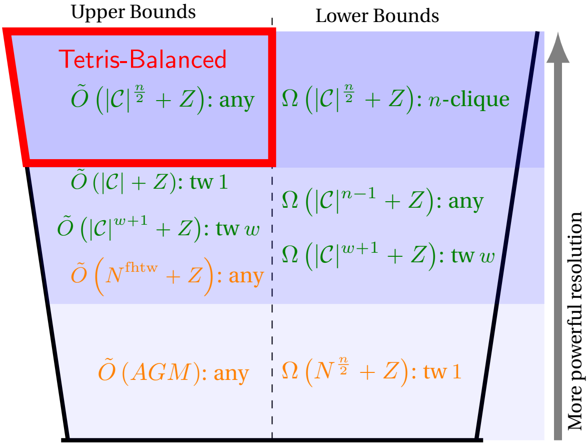
- Algorithm
  1. $\mathcal{C}_{\square} \leftarrow \emptyset$
  2. Fix a dimension ordering
  3. Run Tetris. If an uncovered point $\mathbf{b}$ is found
     - Query for boxes covering $\mathbf{b}$ $\quad (\tilde{O}(1))$
     - Load them into $\mathcal{C}_{\square}$
     - Repeat

Tetris-Reloaded

- Algorithm
  1. $\mathcal{C}_\square \leftarrow \emptyset$
  2. Fix a dimension ordering
  3. Run Tetris. If an uncovered point $\mathbf{b}$ is found
     - Query for boxes covering $\mathbf{b}$ $\qquad (\tilde{O}(1))$
     - Load them into $\mathcal{C}_\square$
     - Repeat
- Analysis
  - $|\mathcal{C}_\square| = \tilde{O}(|\mathcal{C}'_\square|), \quad$ for any $\mathcal{C}'_\square$



Upper Bounds
Lower Bounds
$\tilde{O}\left(|\mathcal{C}|^{\frac{n}{2}} + Z\right)$: any
$\Omega\left(|\mathcal{C}|^{\frac{n}{2}} + Z\right)$: n-clique
$\tilde{O}\left(|\mathcal{C}| + Z\right)$: tw 1
Tetris-Reloaded
$\tilde{O}\left(|\mathcal{C}|^{w+1} + Z\right)$: tw w
$\Omega\left(|\mathcal{C}|^{n-1} + Z\right)$: any
$\Omega\left(|\mathcal{C}|^{w+1} + Z\right)$: tw w
$\tilde{O}\left(N^{\text{fhtw}} + Z\right)$: any
$\tilde{O}\left(AGM\right)$: any
$\Omega\left(N^{\frac{n}{2}} + Z\right)$: tw 1
More powerful resolution
Geometric Resolution
Ordered Geometric Resolution
Tree Ordered Geometric Resolution



Geometric Resolution

Ordered Geometric Resolution

Tree Ordered Geometric Resolution

Table of Contents

Open Problems

- A $\tilde{O}(P)$-algorithm?

Open Problems

- A $\tilde{O}(P)$-algorithm?
  - P is the GEOMETRIC RESOLUTION-proof size

Open Problems

- A $\tilde{O}(P)$-algorithm?
  - P is the GEOMETRIC RESOLUTION-proof size
  - P could be anywhere from $|\mathcal{C}_\square|$ to $\tilde{\Theta}(|\mathcal{C}_\square|^{n/2})$

Open Problems

- A $\tilde{O}(P)$-algorithm?
  - P is the Geometric Resolution-proof size
  - P could be anywhere from $|\mathcal{C}_\square|$ to $\tilde{\Theta}(|\mathcal{C}_\square|^{n/2})$
- A more practical alternative to dyadic encoding?

Open Problems

- A $\tilde{O}(P)$-algorithm?
  - P is the GEOMETRIC RESOLUTION-proof size
  - P could be anywhere from $|\mathcal{C}_\square|$ to $\tilde{\Theta}(|\mathcal{C}_\square|^{n/2})$
- A more practical alternative to dyadic encoding?
  - Shave polylog factors

Open Problems

- A $\tilde{O}(P)$-algorithm?
  - P is the GEOMETRIC RESOLUTION-proof size
  - P could be anywhere from $|\mathcal{C}_\square|$ to $\tilde{\Theta}(|\mathcal{C}_\square|^{n/2})$
- A more practical alternative to dyadic encoding?
  - Shave polylog factors
- Other models for instance optimality?

Open Problems

- A $\tilde{O}(P)$-algorithm?
  - P is the GEOMETRIC RESOLUTION-proof size
  - P could be anywhere from $|\mathcal{C}_\square|$ to $\tilde{\Theta}(|\mathcal{C}_\square|^{n/2})$
- A more practical alternative to dyadic encoding?
  - Shave polylog factors
- Other models for instance optimality?
- A notion of certificates for algebraic algorithms?

Open Problems

- A $\tilde{O}(P)$-algorithm?
  - P is the GEOMETRIC RESOLUTION-proof size
  - P could be anywhere from $|\mathcal{C}_\square|$ to $\tilde{\Theta}(|\mathcal{C}_\square|^{n/2})$
- A more practical alternative to dyadic encoding?
  - Shave polylog factors
- Other models for instance optimality?
- A notion of certificates for algebraic algorithms?
  - Algebraic algorithms can break the $\Omega(|\mathcal{C}_\square|^{n/2})$-lower bound

Open Problems

- A $\tilde{O}(P)$-algorithm?
  - P is the GEOMETRIC RESOLUTION-proof size
  - P could be anywhere from $|\mathcal{C}_\Box|$ to $\tilde{\Theta}(|\mathcal{C}_\Box|^{n/2})$
- A more practical alternative to dyadic encoding?
  - Shave polylog factors
- Other models for instance optimality?
- A notion of certificates for algebraic algorithms?
  - Algebraic algorithms can break the $\Omega(|\mathcal{C}_\Box|^{n/2})$-lower bound
    - e.g. listing triangles in $O(N^{1.408} + N^{1.222} Z^{0.186})$ [Björklund et al, ICALP'14]

Open Problems (cont.)

- Count/Aggregate queries?

Open Problems (cont.)

- Count/Aggregate queries?
  - What is a natural notion of certificates here?

Open Problems (cont.)

- Count/Aggregate queries?
  - What is a natural notion of certificates here?
  - Corresponding instance-optimal algorithm?

Open Problems (cont.)

- Count/Aggregate queries?
  - What is a natural notion of certificates here?
  - Corresponding instance-optimal algorithm?
- Recursive queries?

Open Problems (cont.)

- Count/Aggregate queries?
  - What is a natural notion of certificates here?
  - Corresponding instance-optimal algorithm?
- Recursive queries?
  - e.g. instance-optimal transitive closure?

Open Problems (cont.)

- Count/Aggregate queries?
  - What is a natural notion of certificates here?
  - Corresponding instance-optimal algorithm?
- Recursive queries?
  - e.g. instance-optimal transitive closure?
  - Instance-optimal all-pairs shortest-paths?

Open Problems (cont.)

- Count/Aggregate queries?
  - What is a natural notion of certificates here?
  - Corresponding instance-optimal algorithm?
- Recursive queries?
  - e.g. instance-optimal transitive closure?
  - Instance-optimal all-pairs shortest-paths?
- Instance Optimality under Updates (IVM)?

Many Thanks!
Any Questions/Comments?

Table of Contents

Tetris-Preloaded

- Algorithm
  1. Load all gap boxes
  2. Fix a dimension ordering
  3. Run Tetris

Tetris-Preloaded

- Algorithm
  1. Load all gap boxes
  2. Fix a dimension ordering
  3. Run Tetris
- Underlying Proof System
  - ORDERED GEOMETRIC RESOLUTION

Tetris-Preloaded

- Algorithm
  1. Load all gap boxes
  2. Fix a dimension ordering
  3. Run Tetris
- Underlying Proof System
  - Ordered Geometric Resolution
- Runtime Bounds
  - $\tilde{O}(N + N^{\text{fhtw}} + Z)$
    - $\tilde{O}(N + Z)$ for acyclic queries
    - $\tilde{O}(\mathsf{AGM})$ even without caching (Tree Ordered Geometric Resolution)

Tetris-Reloaded: More details

- Underlying Proof System
  - ORDERED GEOMETRIC RESOLUTION

Tetris-Reloaded: More details

- Underlying Proof System
  - ORDERED GEOMETRIC RESOLUTION
- Runtime Bounds
  - $\tilde{O}(|\mathcal{C}_\square| + Z)$ for treewidth $w = 1$
  - $\tilde{O}(|\mathcal{C}_\square|^{w+1} + Z)$

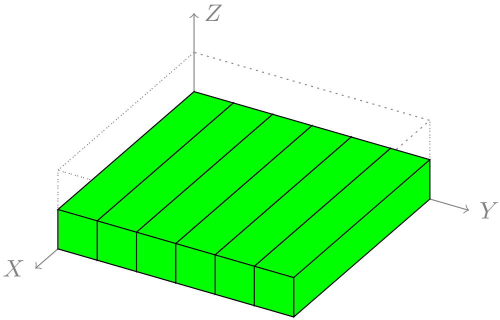
Tetris-Reloaded: More details

- Underlying Proof System
  - ORDERED GEOMETRIC RESOLUTION
- Runtime Bounds
  - $\tilde{O}(|\mathcal{C}_\Box| + Z)$ for treewidth $w = 1$
  - $\tilde{O}(|\mathcal{C}_\Box|^{w+1} + Z)$
- Lower Bounds for ORDERED GEOMETRIC RESOLUTION
  - $\Omega(|\mathcal{C}_\Box| + Z)$ for treewidth $w = 1$
  - $\Omega(|\mathcal{C}_\Box|^{w+1} + Z)$
  - $\Omega(|\mathcal{C}_\Box|^{n-1} + Z)$ for n-clique

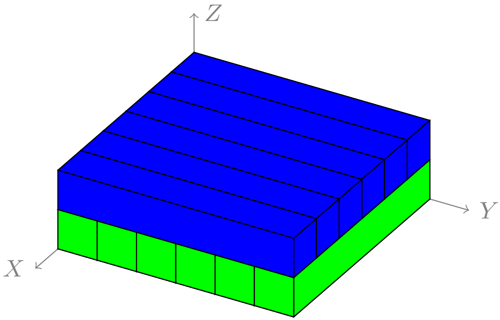
Tetris-Reloaded: More details

- Underlying Proof System
  - ORDERED GEOMETRIC RESOLUTION
- Runtime Bounds
  - $\tilde{O}(|\mathcal{C}_\square| + Z)$ for treewidth $w = 1$
  - $\tilde{O}(|\mathcal{C}_\square|^{w+1} + Z)$
- Lower Bounds for ORDERED GEOMETRIC RESOLUTION
  - $\Omega(|\mathcal{C}_\square| + Z)$ for treewidth $w = 1$
  - $\Omega(|\mathcal{C}_\square|^{w+1} + Z)$
  - $\Omega(|\mathcal{C}_\square|^{n-1} + Z)$ for n-clique
    - *But AGM bound for an n-clique is $\tilde{O}(N^{n/2})$*

$\Omega(|\mathcal{C}_\square|^{n-1})$ for Ordered Geometric Resolution

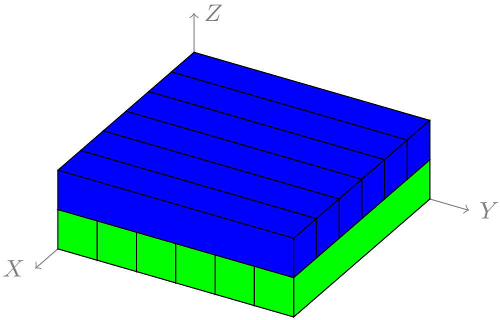


$\Omega(|\mathcal{C}_\square|^{n-1})$ for Ordered Geometric Resolution



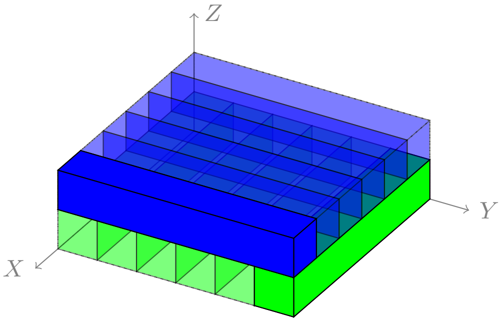
Consider the above certificate $\mathcal{C}_\square$

$\Omega(|\mathcal{C}_\square|^{n-1})$ for Ordered Geometric Resolution



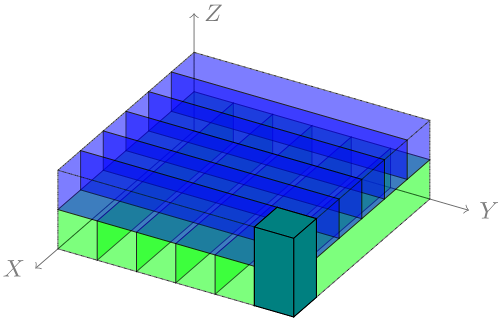
Ordered resolution under any order starting with Z results in $|\mathcal{C}_\square|^2$

$\Omega(|\mathcal{C}_\Box|^{n-1})$ for Ordered Geometric Resolution



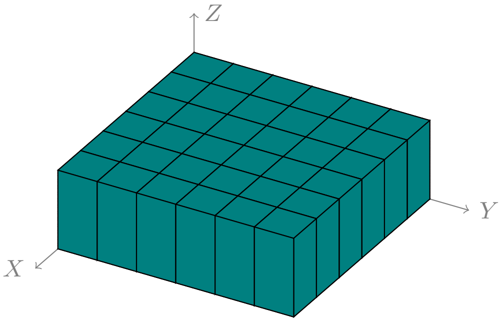
Ordered resolution under any order starting with Z results in $|\mathcal{C}_\Box|^2$

$\Omega(|\mathcal{C}_\square|^{n-1})$ for Ordered Geometric Resolution



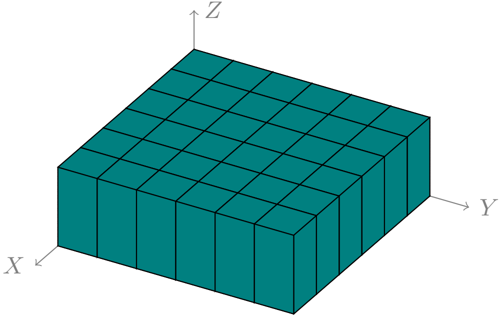
Ordered resolution under any order starting with Z results in $|\mathcal{C}_\square|^2$

$\Omega(|\mathcal{C}_\square|^{n-1})$ for Ordered Geometric Resolution



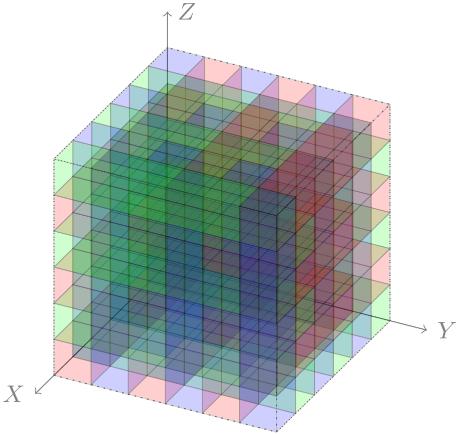
Ordered resolution under any order starting with Z results in $|\mathcal{C}_\square|^2$

$\Omega(|\mathcal{C}_\square|^{n-1})$ for Ordered Geometric Resolution



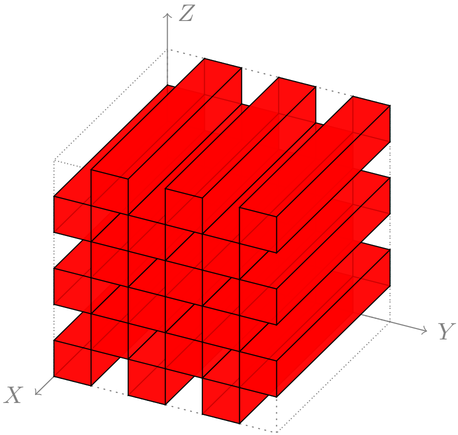
By concatenating together 3 rotated instances of the above, we get a lower bound of $|\mathcal{C}_\square|^2$ for any fixed order

$\Omega(|\mathcal{C}_{\Box}|^{n/2})$ for (Unordered) Geometric Resolution



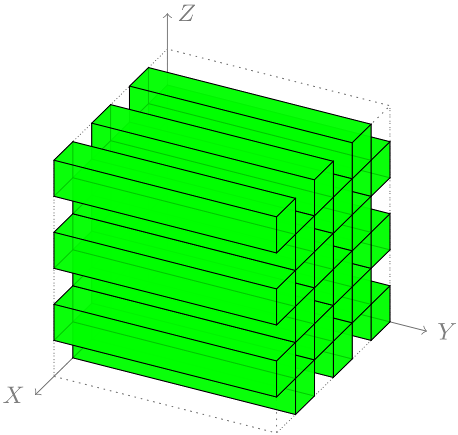
- Let $m := \sqrt{|\mathcal{C}_{\Box}|/3}$

$\Omega(|\mathcal{C}_{\Box}|^{n/2})$ for (Unordered) Geometric Resolution



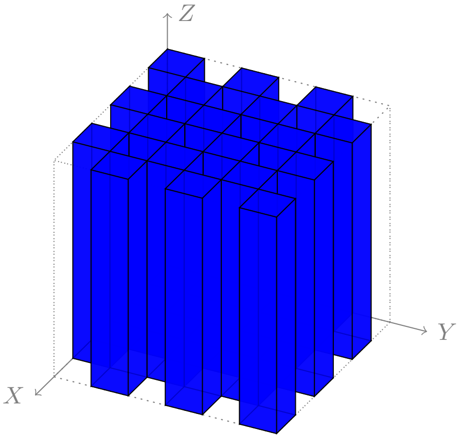
- Let $m := \sqrt{|\mathcal{C}_{\Box}|/3}$
- $m \times m$ red boxes

$\Omega(|\mathcal{C}_\square|^{n/2})$ for (Unordered) GEOMETRIC RESOLUTION



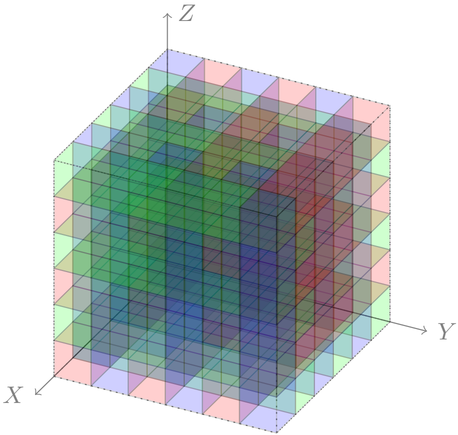
- Let $m := \sqrt{|\mathcal{C}_\square|/3}$
- $m \times m$ red boxes
- $m \times m$ green boxes

$\Omega(|\mathcal{C}_\square|^{n/2})$ for (Unordered) GEOMETRIC RESOLUTION



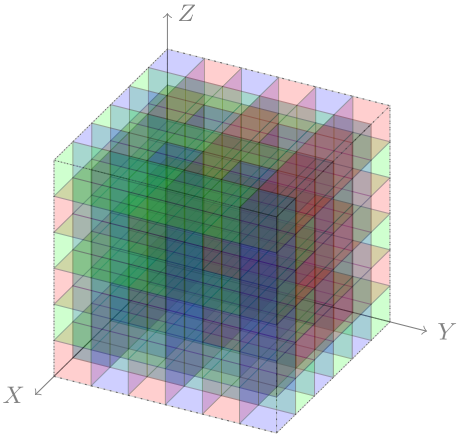
- Let $m := \sqrt{|\mathcal{C}_\square|/3}$
- $m \times m$ red boxes
- $m \times m$ green boxes
- $m \times m$ blue boxes

$\Omega(|\mathcal{C}_\square|^{n/2})$ for (Unordered) Geometric Resolution



- Let $m := \sqrt{|\mathcal{C}_\square|/3}$
- $m \times m$ red boxes
- $m \times m$ green boxes
- $m \times m$ blue boxes

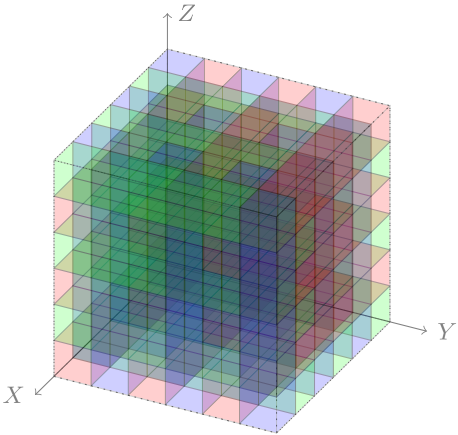
$\Omega(|\mathcal{C}_\square|^{n/2})$ for (Unordered) Geometric Resolution



- Let $m := \sqrt{|\mathcal{C}_\square|/3}$
- $m \times m$ red boxes
- $m \times m$ green boxes
- $m \times m$ blue boxes

Resolving any two boxes results in a box of size 2

$\Omega(|\mathcal{C}_\square|^{n/2})$ for (Unordered) GEOMETRIC RESOLUTION



- Let $m := \sqrt{|\mathcal{C}_\square|/3}$
- $m \times m$ red boxes
- $m \times m$ green boxes
- $m \times m$ blue boxes

Resolving any two boxes results in a box of size 2
(This does NOT prove the lower bound! Just for intuition..)

Tetris-Balanced

- Algorithm

Tetris-Balanced

- Algorithm
  1. Suppose the input space has n-dimensions $A_1, ..., A_n$.

Tetris-Balanced

- Algorithm
  1. Suppose the input space has n-dimensions $A_1, \ldots, A_n$.
     - For each $i \in \{1, \ldots, n-2\}$
       "Split" dimension A_i into two smaller dimensions (A'_i, A''_i) in a "balanced" way.

Tetris-Balanced

- Algorithm
  1. Suppose the input space has n-dimensions $A_1, ..., A_n$.
     - For each $i \in \{1, ..., n-2\}$
       "Split" dimension A_i into two smaller dimensions (A_i', A_i'') in a "balanced" way.
  2. Fix the dimension order:

$$(A_1', A_2', \ldots, A_{n-2}', A_{n-1}, A_n, A_{n-2}'', A_{n-3}'', \ldots, A_1'')$$

Tetris-Balanced

- Algorithm
  1. Suppose the input space has n-dimensions $A_1, ..., A_n$.
     - For each $i \in \{1, ..., n-2\}$
       "Split" dimension A_i into two smaller dimensions (A_i', A_i'') in a "balanced" way.
  2. Fix the dimension order:

$$(A_1', A_2', \ldots, A_{n-2}', A_{n-1}, A_n, A_{n-2}'', A_{n-3}'', \ldots, A_1'')$$

  3. Run Tetris-Reloaded (in the new space of dimension $2n-2$)

Tetris-Balanced

- Algorithm
  1. Suppose the input space has n-dimensions $A_1, ..., A_n$.
     - For each $i \in \{1, ..., n-2\}$
       "Split" dimension A_i into two smaller dimensions (A_i', A_i'') in a "balanced" way.
  2. Fix the dimension order:

$$(A_1', A_2', \ldots, A_{n-2}', A_{n-1}, A_n, A_{n-2}'', A_{n-3}'', \ldots, A_1'')$$

  3. Run Tetris-Reloaded (in the new space of dimension $2n-2$)
- Underlying Proof System

Tetris-Balanced

- Algorithm
  1. Suppose the input space has n-dimensions $A_1, ..., A_n$.
     - For each $i \in \{1, ..., n-2\}$
       "Split" dimension A_i into two smaller dimensions (A_i', A_i'') in a "balanced" way.
  2. Fix the dimension order:

$$(A_1', A_2', \ldots, A_{n-2}', A_{n-1}, A_n, A_{n-2}'', A_{n-3}'', \ldots, A_1'')$$

  3. Run Tetris-Reloaded (in the new space of dimension $2n-2$)
- Underlying Proof System
  - GEOMETRIC RESOLUTION

Tetris-Balanced

- Algorithm
  1. Suppose the input space has n-dimensions $A_1, ..., A_n$.
     - For each $i \in \{1, ..., n-2\}$
       "Split" dimension A_i into two smaller dimensions (A'_i, A''_i) in a "balanced" way.
  2. Fix the dimension order:

$$(A'_1, A'_2, \ldots, A'_{n-2}, A_{n-1}, A_n, A''_{n-2}, A''_{n-3}, \ldots, A''_1)$$

  3. Run Tetris-Reloaded (in the new space of dimension $2n-2$)
- Underlying Proof System
  - Geometric Resolution
- Runtime Bound

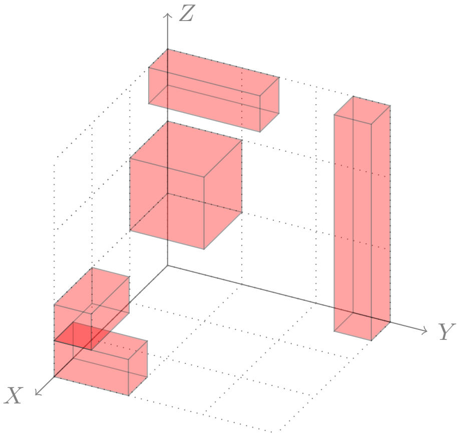
Tetris-Balanced

- Algorithm
  1. Suppose the input space has n-dimensions $A_1, ..., A_n$.
     - For each $i \in \{1, ..., n-2\}$
       "Split" dimension A_i into two smaller dimensions (A_i', A_i'') in a "balanced" way.
  2. Fix the dimension order:

$$(A_1', A_2', \ldots, A_{n-2}', A_{n-1}, A_n, A_{n-2}'', A_{n-3}'', \ldots, A_1'')$$

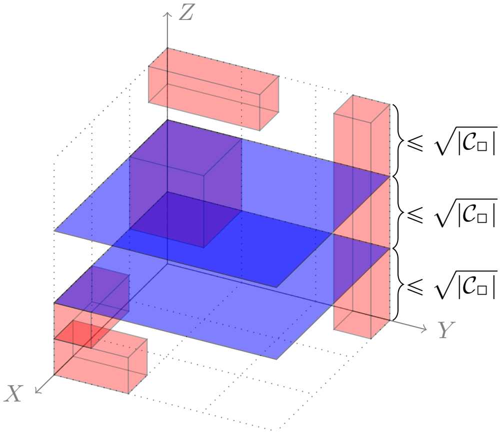
  3. Run Tetris-Reloaded (in the new space of dimension $2n-2$)
- Underlying Proof System
  - GEOMETRIC RESOLUTION
- Runtime Bound
  - $\tilde{O}(|\mathcal{C}_\square|^{n/2} + Z)$

Tetris-Balanced: Splitting Explained



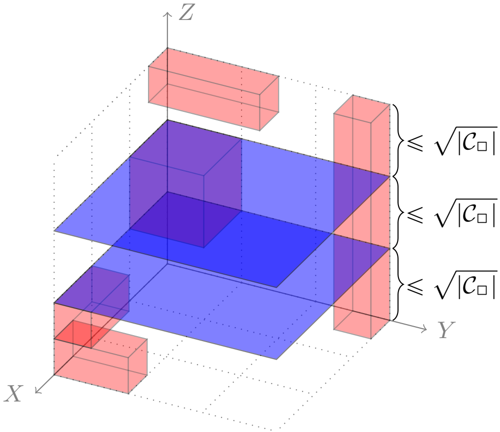
Consider the above box set $\mathcal{C}_{\square}$

Tetris-Balanced: Splitting Explained



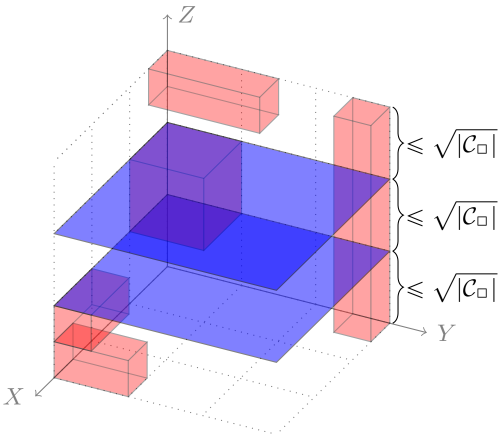
Split Z into $\sqrt{|\mathcal{C}_\square|}$ slices where each slice has $\sqrt{|\mathcal{C}_\square|}$ boxes fully contained in the slice

Tetris-Balanced: Splitting Explained



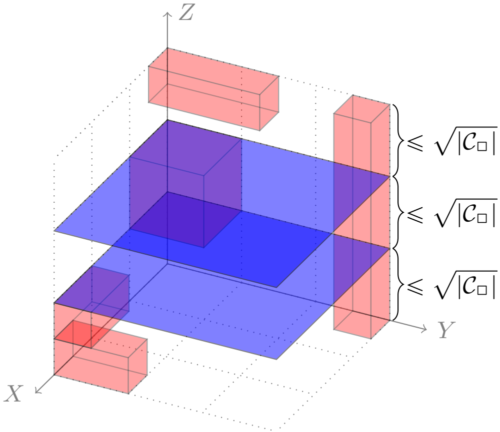
Do resolution over Z only within each slice

Tetris-Balanced: Splitting Explained



Then resolve over X and Y

Tetris-Balanced: Splitting Explained



Then resolve the slices together over Z

Some Followup Works

- K. Alway, "Domain Ordering and Box Cover Problems for Beyond Worst-Case Join Processing", Master Thesis, Waterloo, 2019.
  - Given a relation R with N tuples, generate all maximal dyadic gap boxes of R in time $\tilde{O}(N)$.
  - Strengthens the notion of $\mathcal{C}_{\square}$.
- J. Dobler, and A. Rudra, "Implementation of Tetris as a Model Counter", ArXiV 2017.