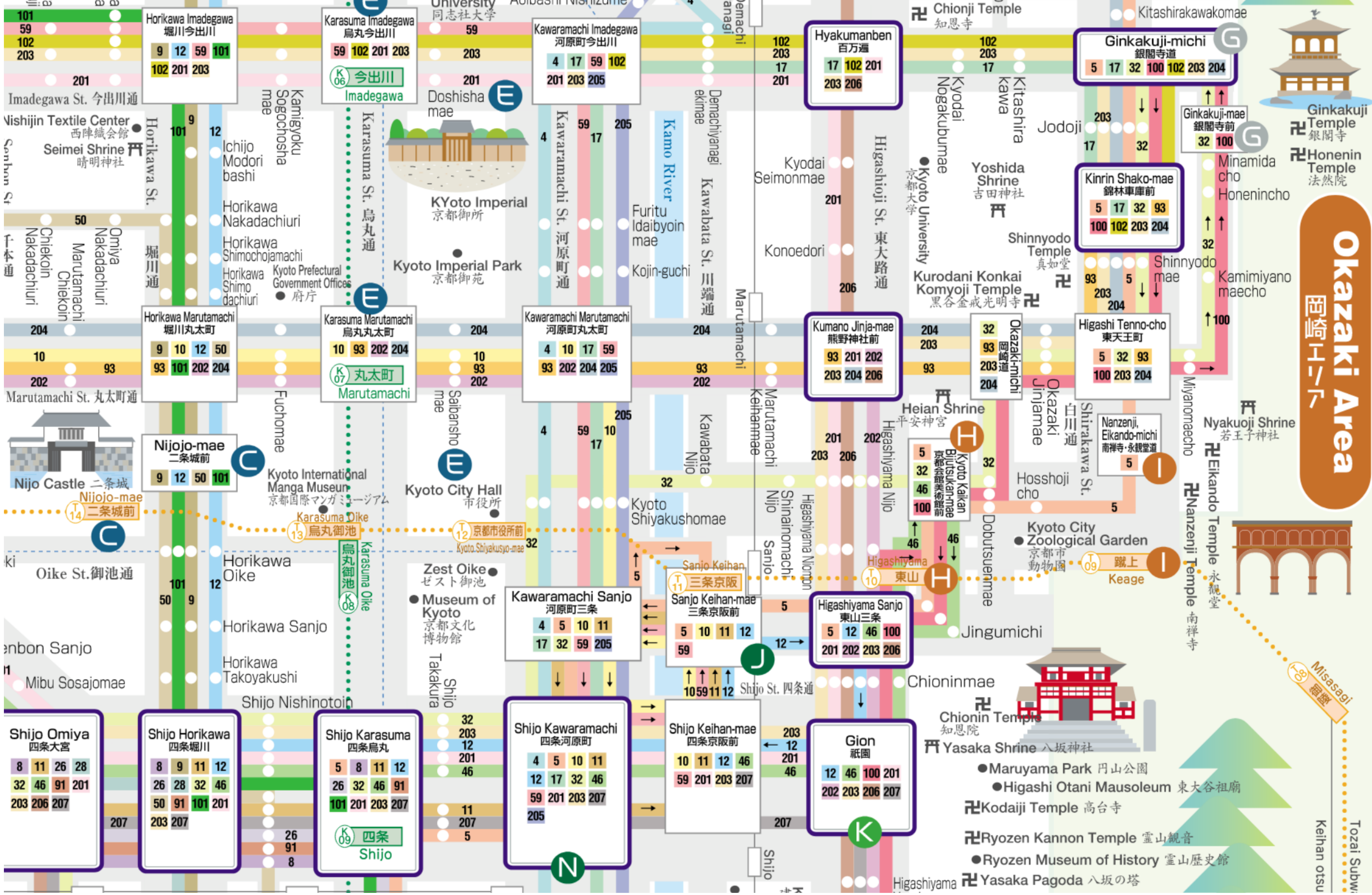
Parameterized Approaches to Orthogonal Compaction

Walter Didimo Siddharth Gupta Philipp Kindermann
Giuseppe Liotta *Alexander Wolff* Meirav Zehavi

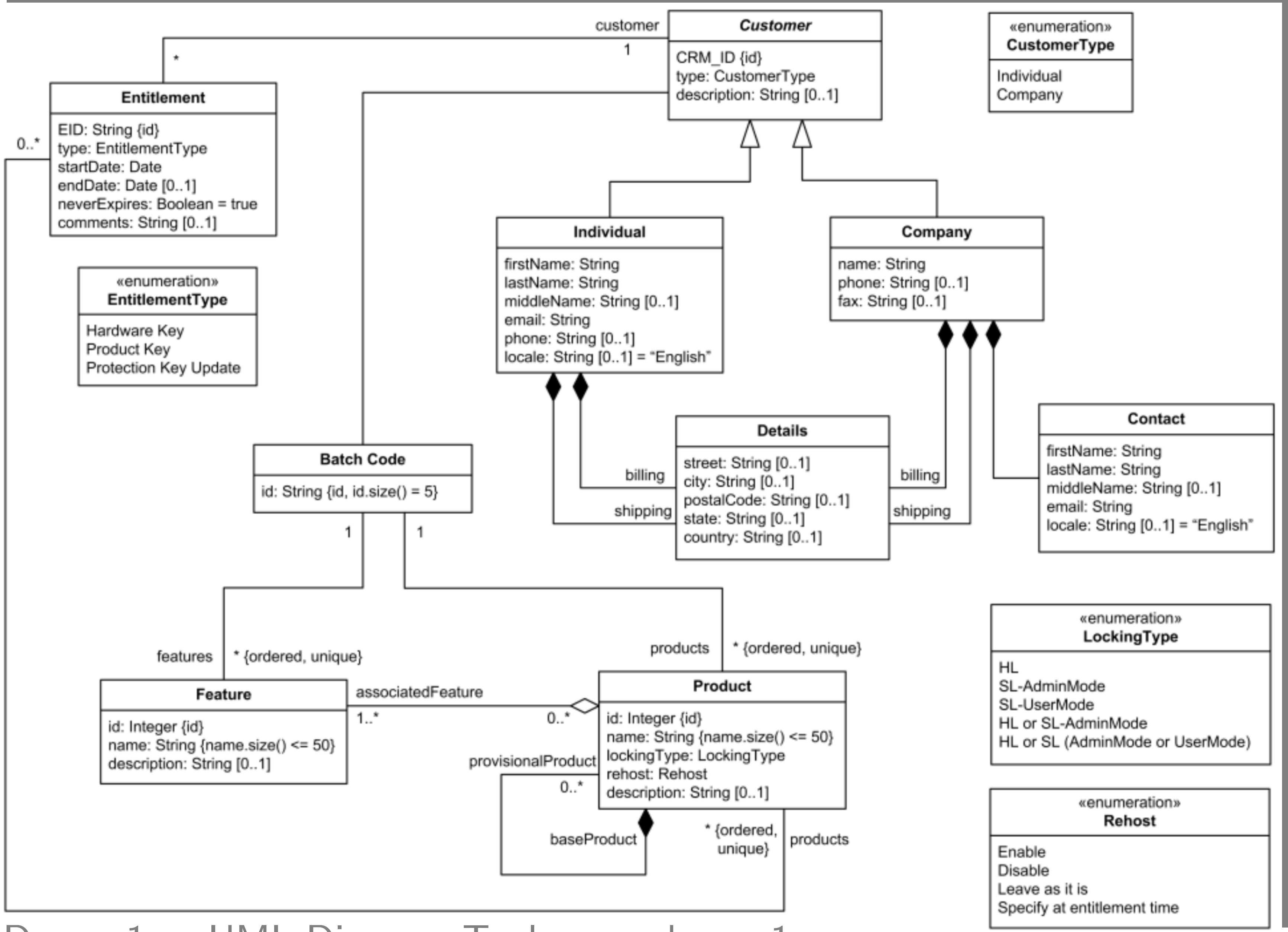
SOFSEM 2023

Orthogonal Graph Drawing

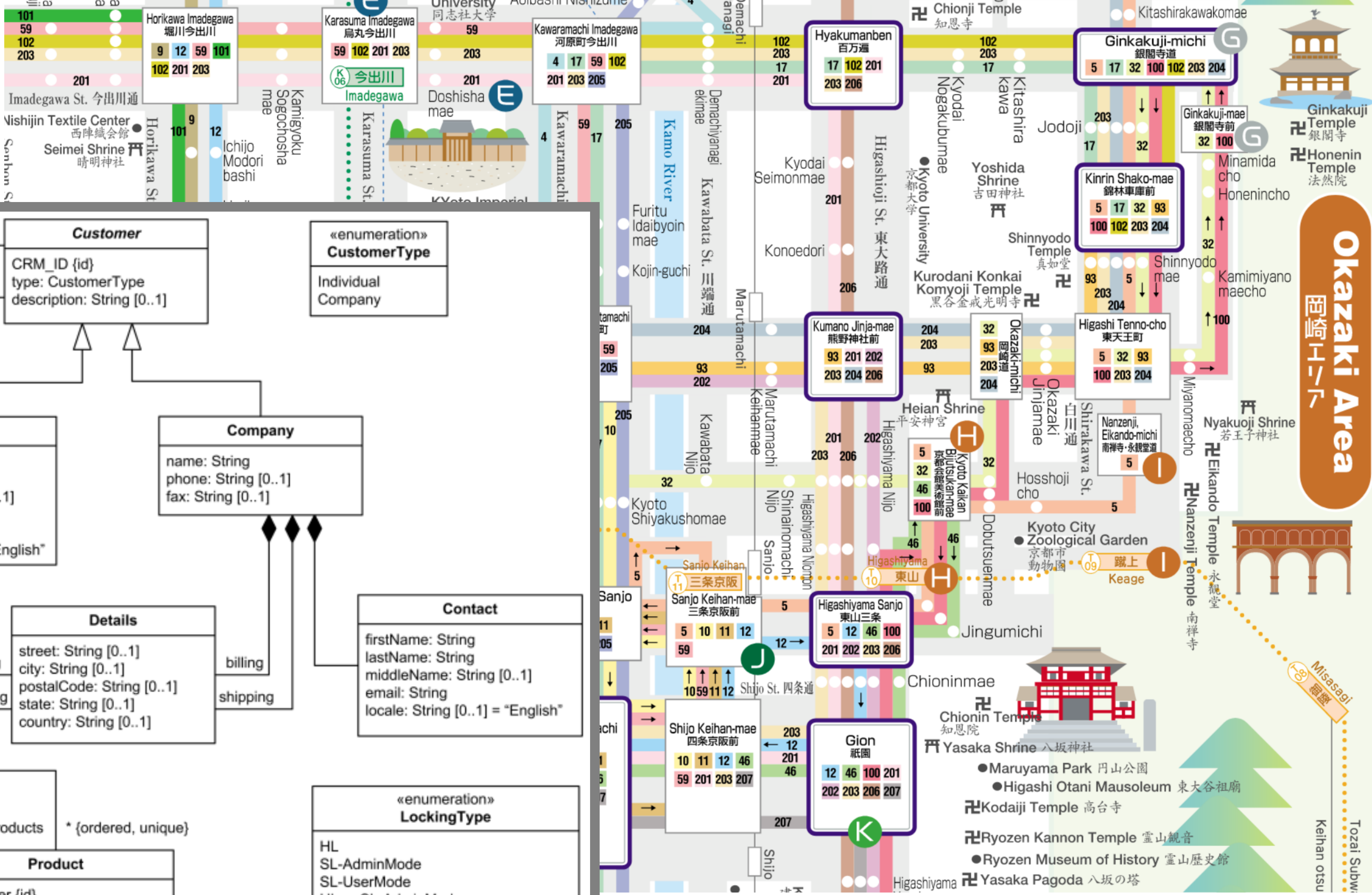


Kyoto Transportation Authority (c) 2008

Orthogonal Graph Drawing

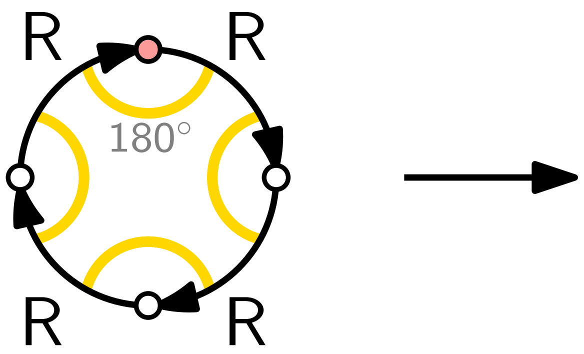


Dragon1 as UML Diagram Tool: www.dragon1.com

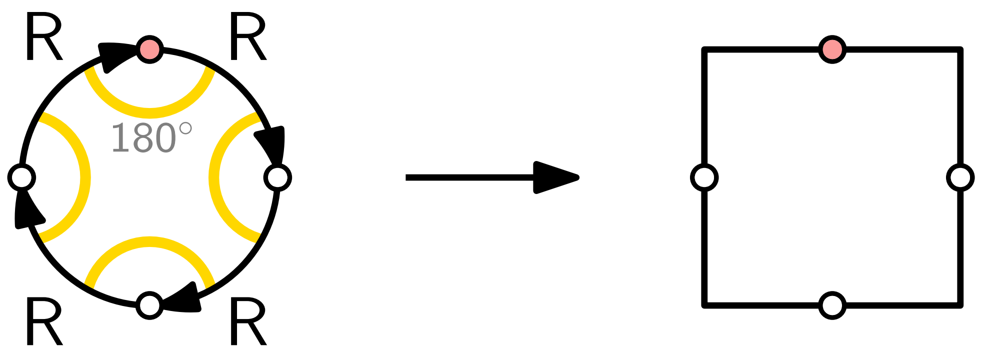


Kyoto Transportation Authority (c) 2008

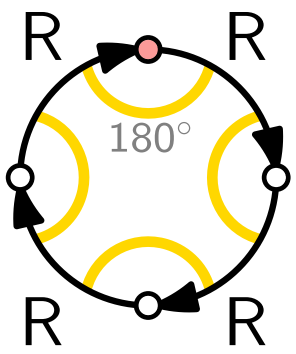
Orthogonal Representation vs. Orthogonal Drawing

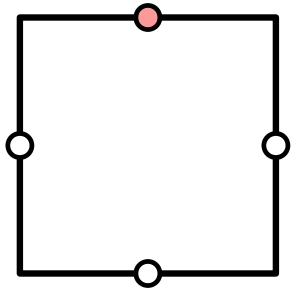


Orthogonal Representation vs. Orthogonal Drawing

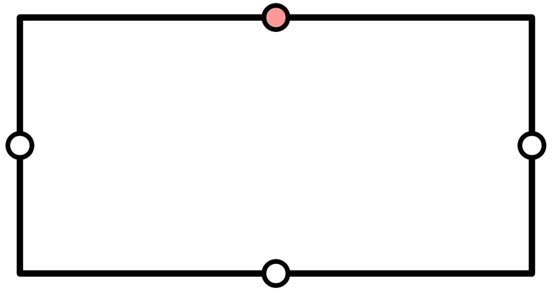


Orthogonal Representation vs. Orthogonal Drawing

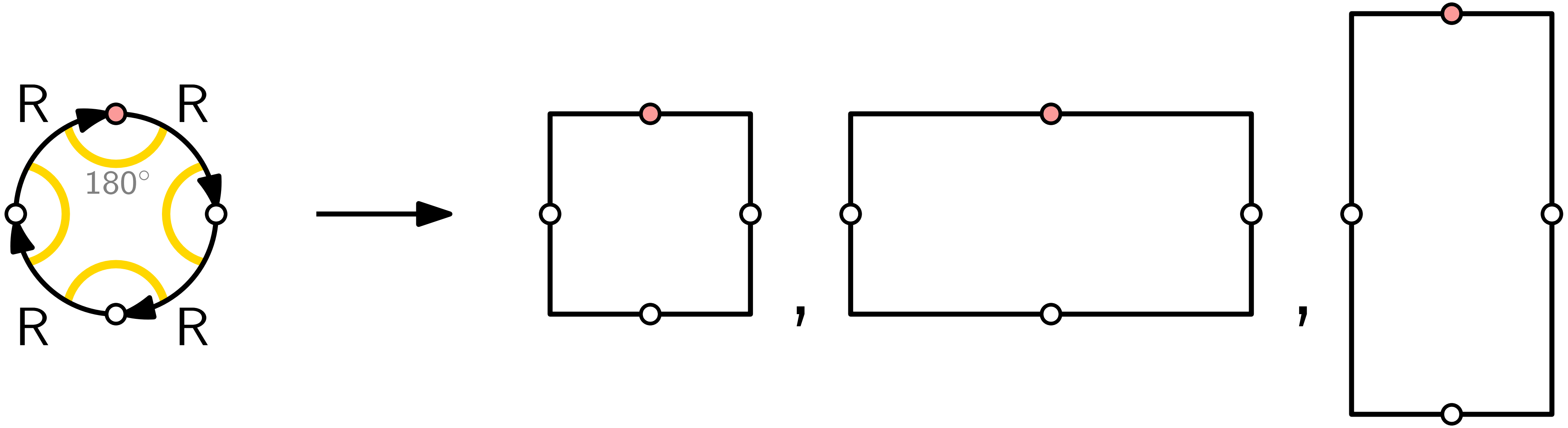




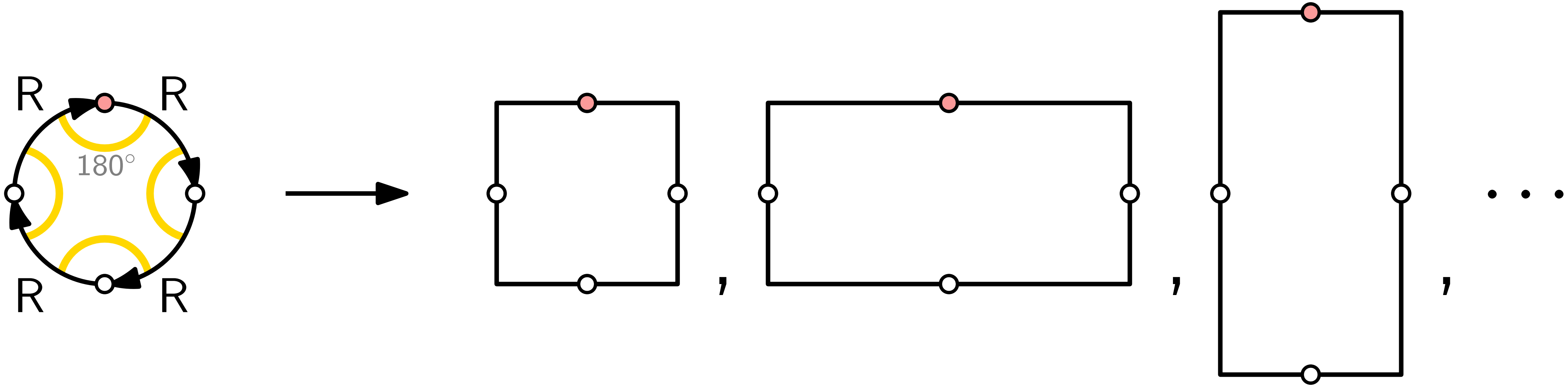
,



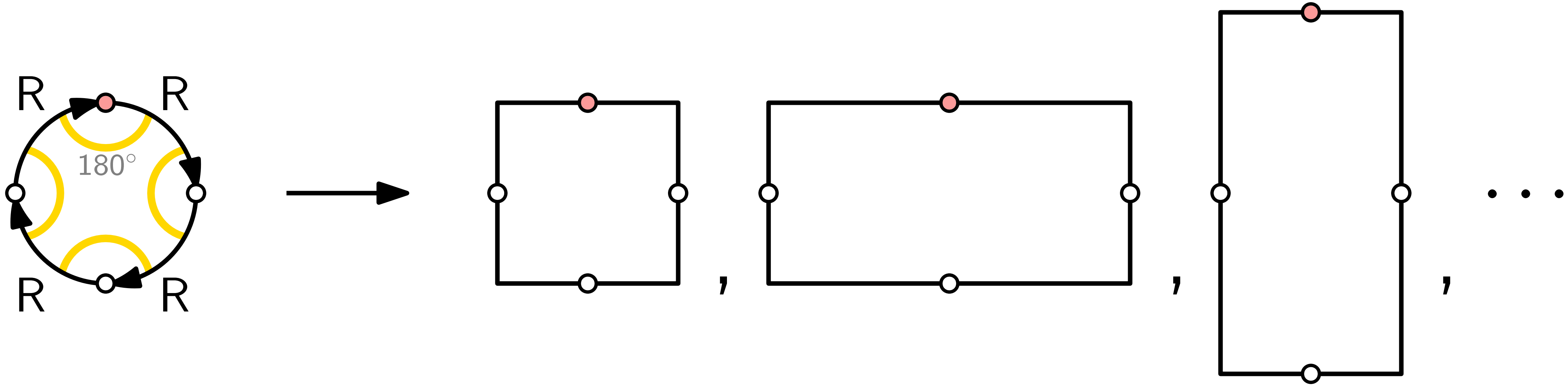
Orthogonal Representation vs. Orthogonal Drawing

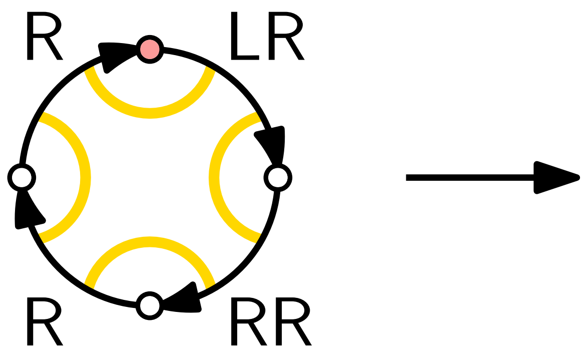


Orthogonal Representation vs. Orthogonal Drawing

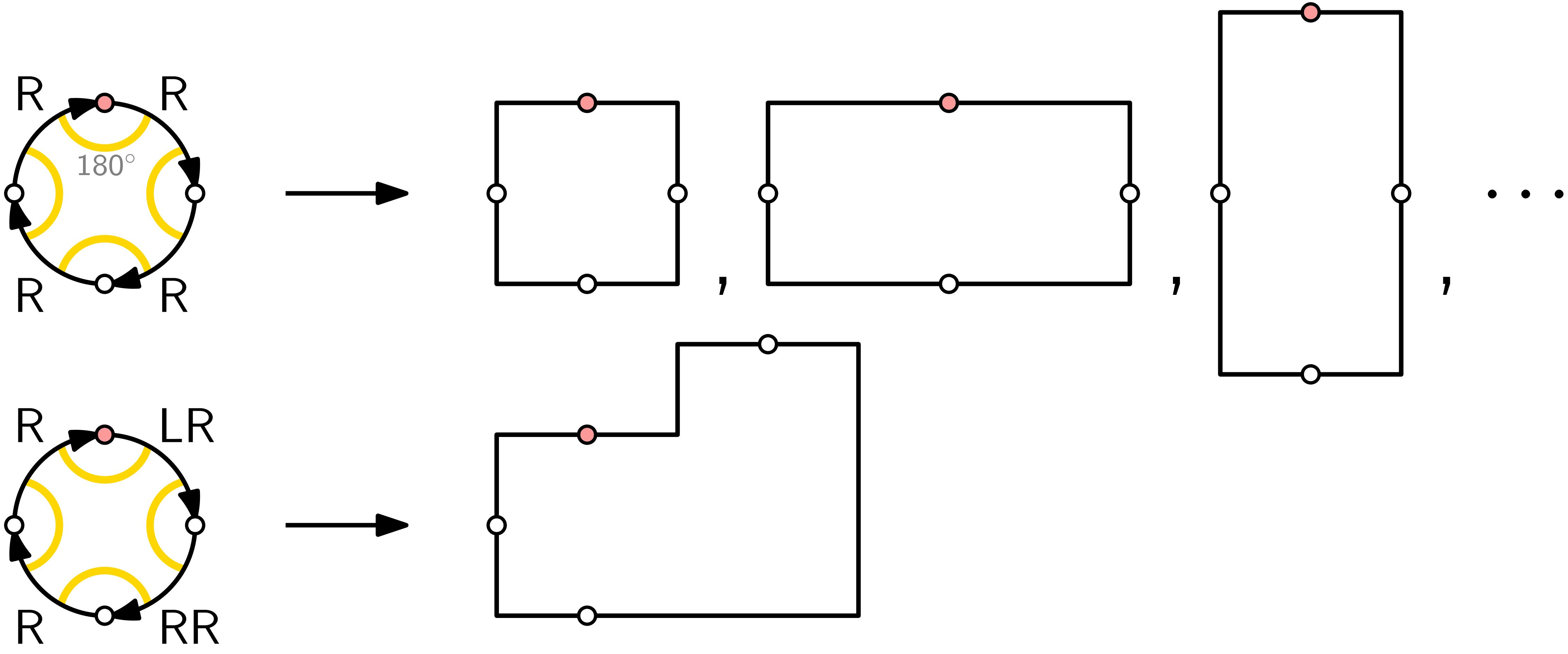


Orthogonal Representation vs. Orthogonal Drawing

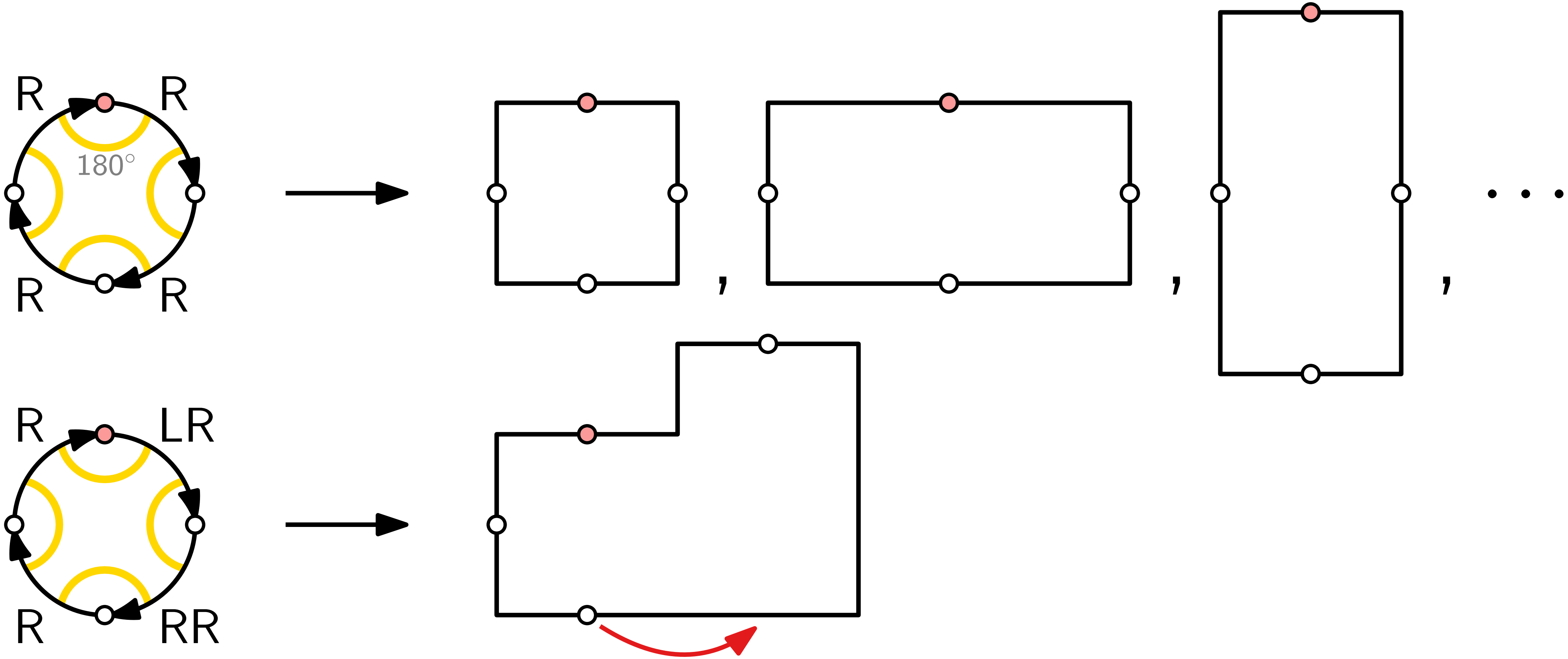




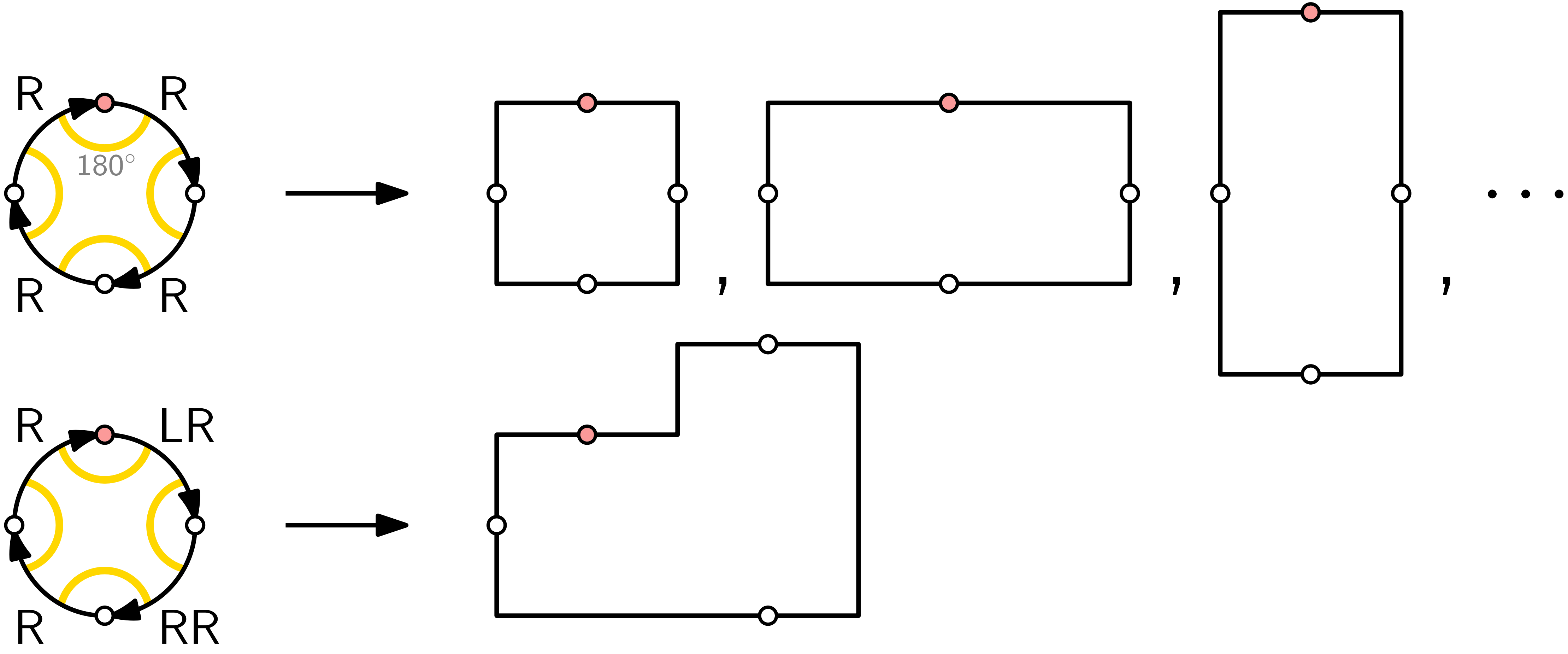
Orthogonal Representation vs. Orthogonal Drawing



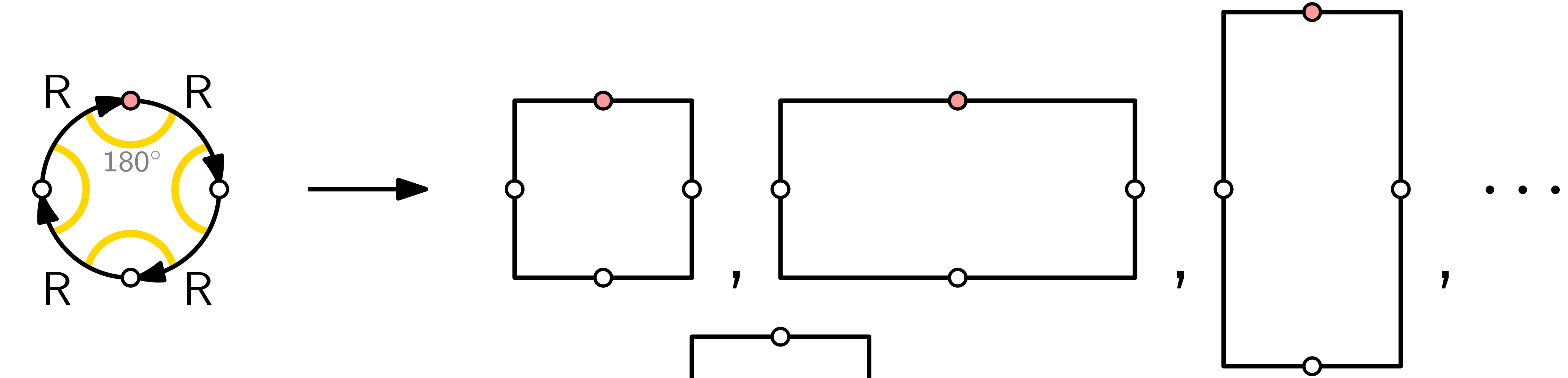
Orthogonal Representation vs. Orthogonal Drawing

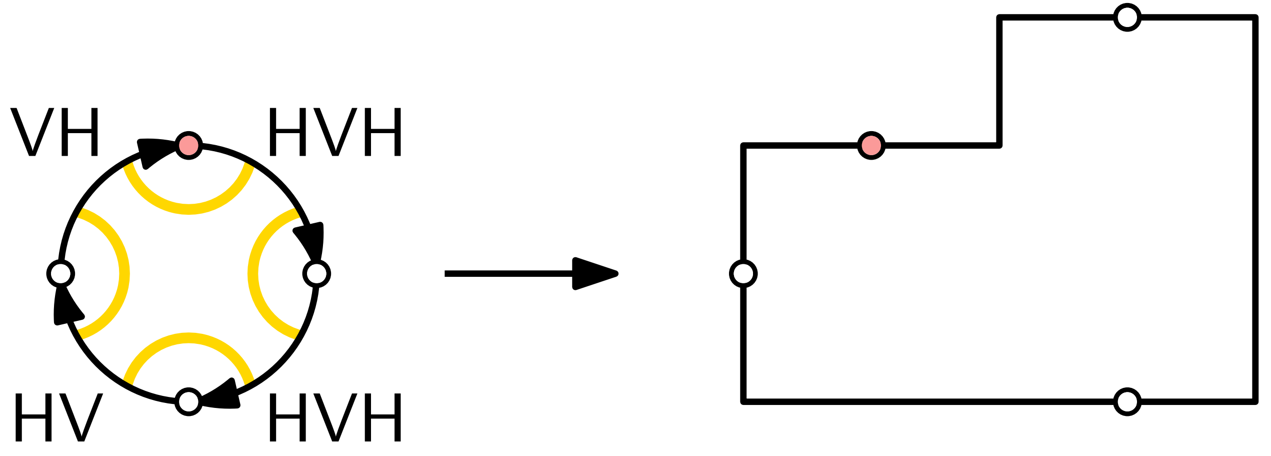


Orthogonal Representation vs. Orthogonal Drawing



Orthogonal Representation vs. Orthogonal Drawing





Previous Work

- Tamassia's seminal work [SIAM J Comp 1987]: Given *embedded* graph of max-degree 4, can compute orthogonal representation with minimum number of bends (via flow netw.).

Previous Work

- Tamassia's seminal work [SIAM J Comp 1987]: Given *embedded* graph of max-degree 4, can compute orthogonal representation with minimum number of bends (via flow netw.).

- *Orthogonal Compaction* (OC) is NP-hard. In other words:
  Given an orthogonal representation R, it is NP-hard to find a minimum-area drawing of R.
  [Patrignani: CGTA 2001]

Previous Work

- Tamassia's seminal work [SIAM J Comp 1987]: Given *embedded* graph of max-degree 4, can compute orthogonal representation with minimum number of bends (via flow netw.).

- *Orthogonal Compaction* (OC) is NP-hard. In other words:
  Given an orthogonal representation R, it is NP-hard to find a minimum-area drawing of R.
  [Patrignani: CGTA 2001]

- Even if the given graph is just a cycle :-(
  [Evans, Fleszar, Kindermann, Saeedi, Shin, Wolff: CGTA 2022]

Previous Work

- Tamassia's seminal work [SIAM J Comp 1987]: Given *embedded* graph of max-degree 4, can compute orthogonal representation with minimum number of bends (via flow netw.).

- *Orthogonal Compaction* (OC) is NP-hard. In other words: Given an orthogonal representation R, it is NP-hard to find a minimum-area drawing of R.
  [Patrignani: CGTA 2001]

- Even if the given graph is just a cycle :-(
  [Evans, Fleszar, Kindermann, Saeedi, Shin, Wolff: CGTA 2022]

- Cycles have pathwidth (and treewidth) 2, so cannot expect FPT algorithm for OC w.r.t. these parameters :-(

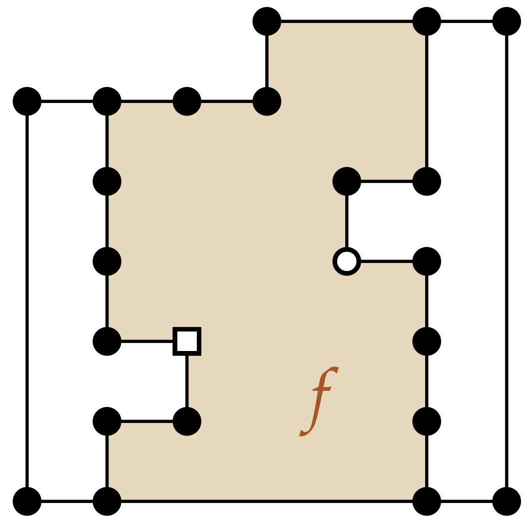
Kitty Corners

What's so hard about OC then??

Kitty Corners

What's so hard about OC then??

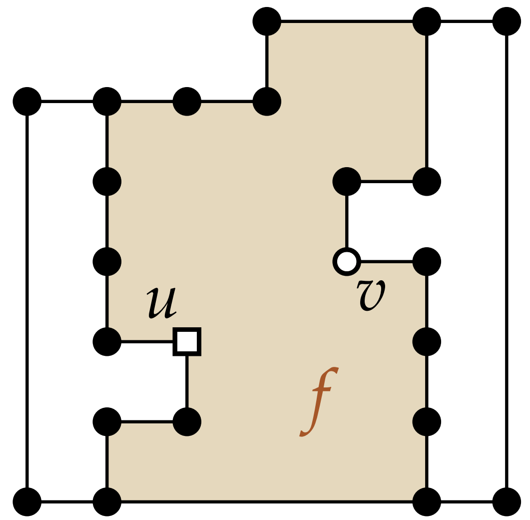
Let f be a face of G.



Kitty Corners

What's so hard about OC then??

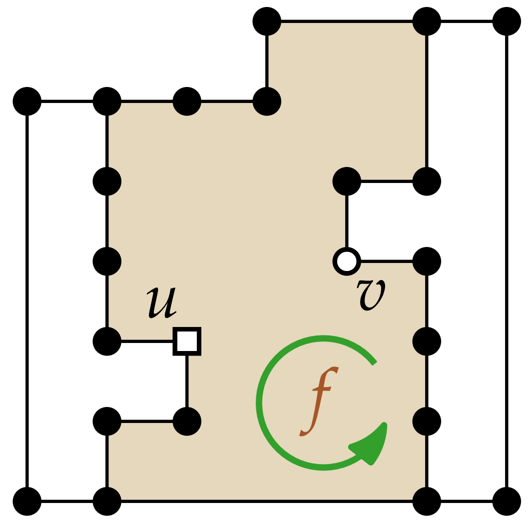
Let f be a face of G. Let u and v be two reflex vertices of f.



Kitty Corners

What's so hard about OC then??

Let f be a face of G. Let u and v be two reflex vertices of f. We direct f ccw iff f is internal.

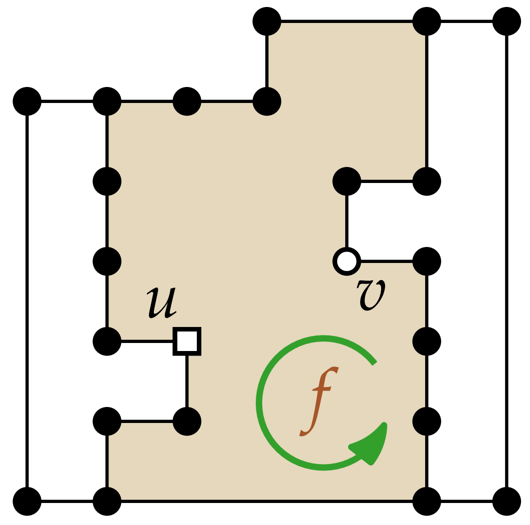


Kitty Corners

What's so hard about OC then??

Let f be a face of G. Let u and v be two reflex vertices of f. We direct f ccw iff f is internal.

rot$(u, v) := \#$ convex corners $- \#$ reflex corners on ∂f from u (included) to v (excluded);
a reflex vertex of degree 1 is counted like two reflex vertices.

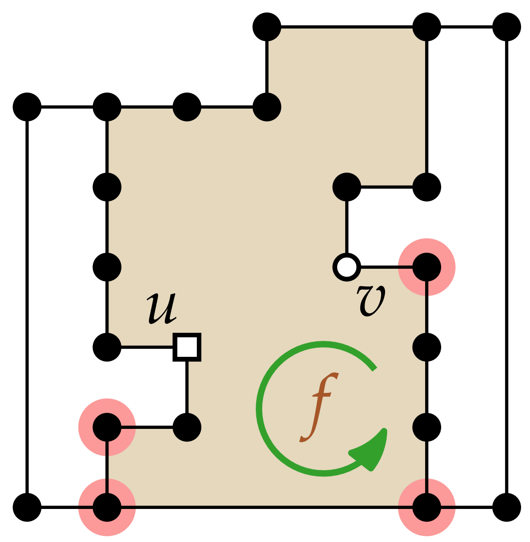


Kitty Corners

What's so hard about OC then??

Let f be a face of G. Let u and v be two reflex vertices of f. We direct f ccw iff f is internal.

$\text{rot}(u, v) :=$ # convex corners − # reflex corners on ∂f from u (included) to v (excluded);
a reflex vertex of degree 1 is counted like two reflex vertices.

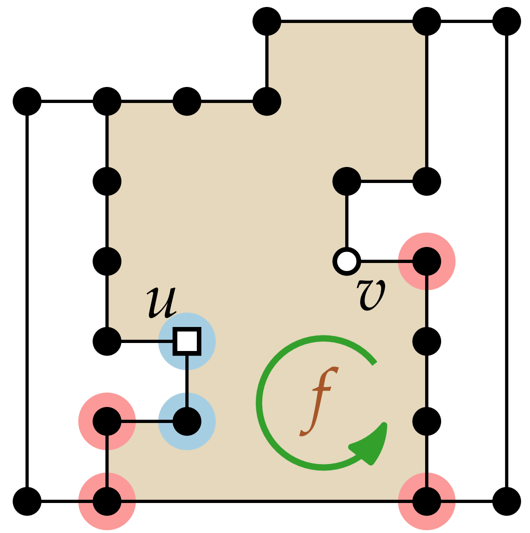


Kitty Corners

What's so hard about OC then??

Let f be a face of G. Let u and v be two reflex vertices of f. We direct f ccw iff f is internal.

$\mathrm{rot}(u, v) :=$ # convex corners $-$ # reflex corners on ∂f from u (included) to v (excluded);
a reflex vertex of degree 1 is counted like two reflex vertices.



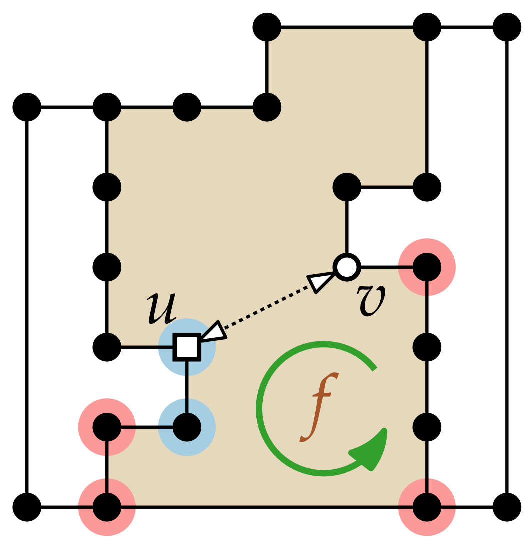
Kitty Corners

What's so hard about OC then??

Let f be a face of G. Let u and v be two reflex vertices of f. We direct f ccw iff f is internal.

$\text{rot}(u, v) :=$ # convex corners $-$ # reflex corners on ∂f from u (included) to v (excluded);
a reflex vertex of degree 1 is counted like two reflex vertices.

We say that $\{u, v\}$ is a *pair of kitty corners* of f
if $\text{rot}(u, v) = 2$ or $\text{rot}(v, u) = 2$.



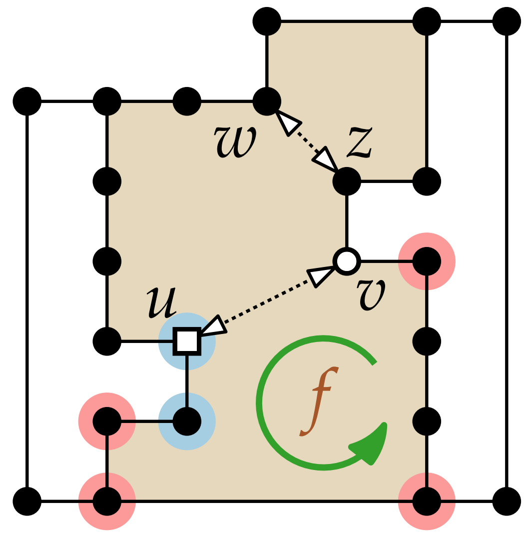
Kitty Corners

What's so hard about OC then??

Let f be a face of G. Let u and v be two reflex vertices of f. We direct f ccw iff f is internal.

$\text{rot}(u, v) :=$ # convex corners $-$ # reflex corners on ∂f from u (included) to v (excluded);
a reflex vertex of degree 1 is counted like two reflex vertices.

We say that $\{u, v\}$ is a *pair of kitty corners* of f
if $\text{rot}(u, v) = 2$ or $\text{rot}(v, u) = 2$.



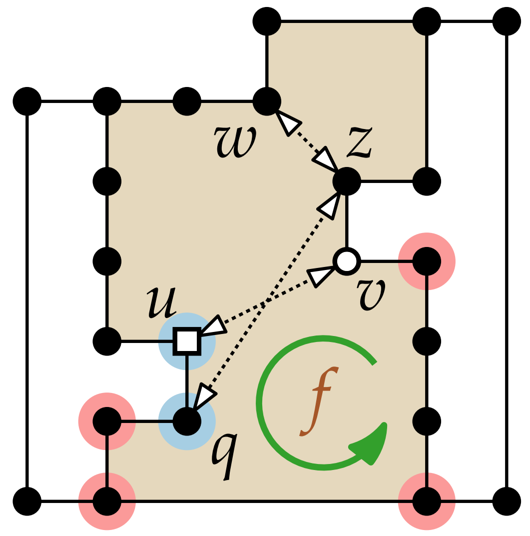
Kitty Corners

What's so hard about OC then??

Let f be a face of G. Let u and v be two reflex vertices of f. We direct f ccw iff f is internal.

$\text{rot}(u, v) :=$ # convex corners $-$ # reflex corners on ∂f from u (included) to v (excluded);
a reflex vertex of degree 1 is counted like two reflex vertices.

We say that $\{u, v\}$ is a *pair of kitty corners* of f
if $\text{rot}(u, v) = 2$ or $\text{rot}(v, u) = 2$.



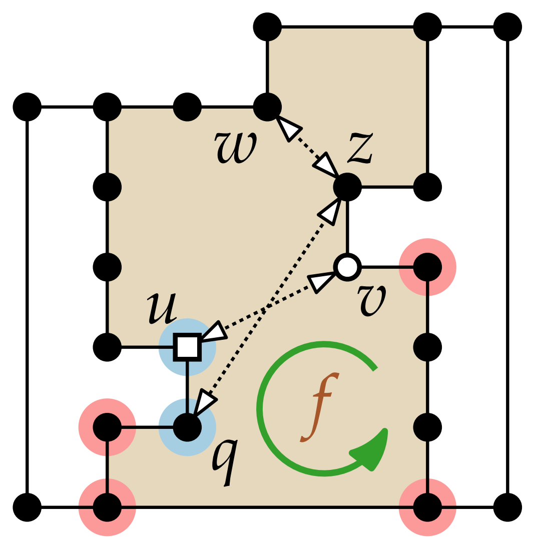
Kitty Corners

What's so hard about OC then??

Let f be a face of G. Let u and v be two reflex vertices of f. We direct f ccw iff f is internal.

$\mathrm{rot}(u, v) :=$ # convex corners $-$ # reflex corners on ∂f from u (included) to v (excluded);
a reflex vertex of degree 1 is counted like two reflex vertices.

We say that $\{u, v\}$ is a *pair of kitty corners* of f
if $\mathrm{rot}(u, v) = 2$ or $\mathrm{rot}(v, u) = 2$.



A vertex is a *kitty corner* if it is part of a pair of kitty corners.

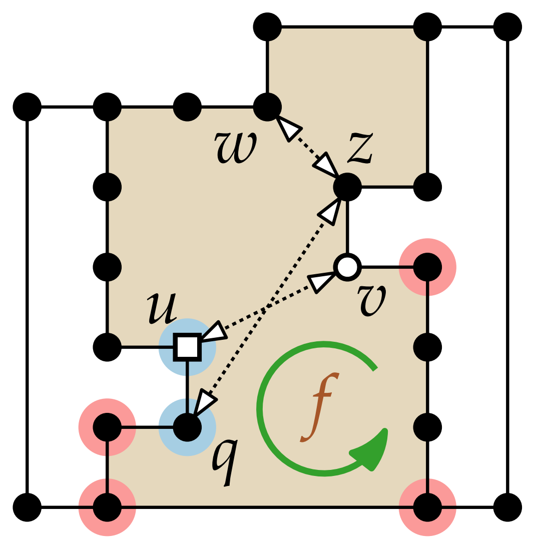
Kitty Corners

What's so hard about OC then??

Let f be a face of G. Let u and v be two reflex vertices of f. We direct f ccw iff f is internal.

$\text{rot}(u, v) :=$ # convex corners $-$ # reflex corners on ∂f from u (included) to v (excluded);
a reflex vertex of degree 1 is counted like two reflex vertices.

We say that $\{u, v\}$ is a *pair of kitty corners* of f
if $\text{rot}(u, v) = 2$ or $\text{rot}(v, u) = 2$.



A vertex is a *kitty corner* if it is part of a pair of kitty corners.

An orthogonal representation R is *turn-regular* if R has no kitty corners.

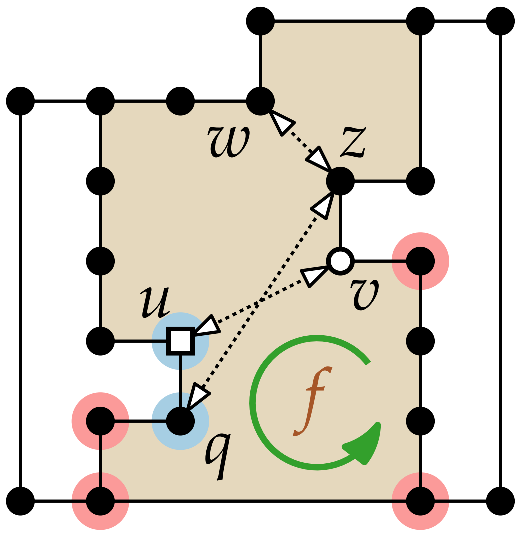
Kitty Corners

What's so hard about OC then??

Let f be a face of G. Let u and v be two reflex vertices of f. We direct f ccw iff f is internal.

rot$(u, v) :=$ # convex corners $-$ # reflex corners on ∂f from u (included) to v (excluded);
a reflex vertex of degree 1 is counted like two reflex vertices.

We say that $\{u, v\}$ is a *pair of kitty corners* of f
if rot$(u, v) = 2$ or rot$(v, u) = 2$.

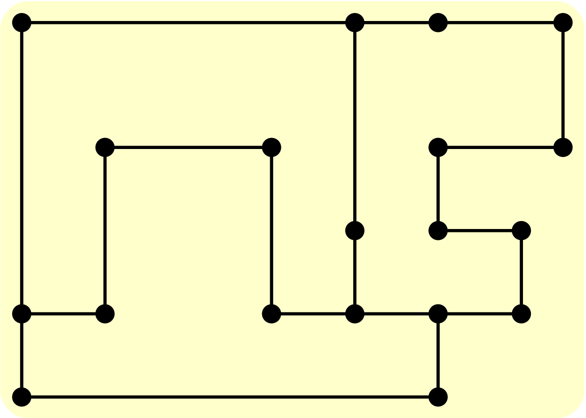


A vertex is a *kitty corner* if it is part of a pair of kitty corners.

An orthogonal representation R is *turn-regular* if R has no kitty corners.

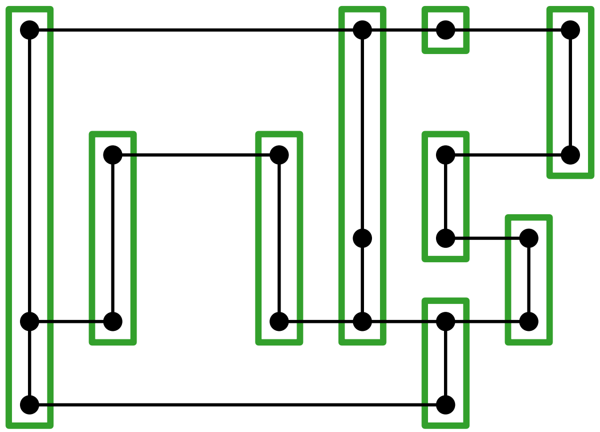
Theorem. If R is turn-regular, a min-area drawing of R can be computed in linear time.
[Bridgeman, Di Battista, Didimo, Liotta, Tamassia, Vismara: CGTA 2000]

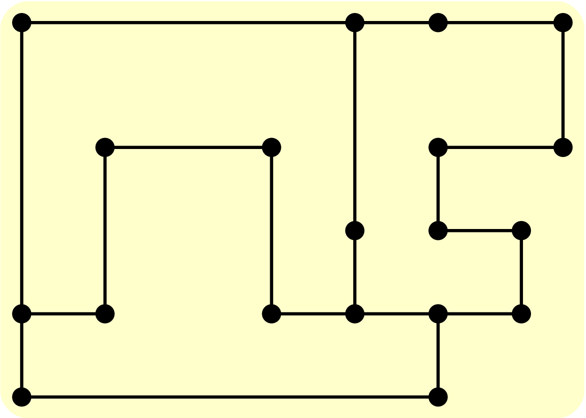
Drawing Turn-Regular Representations Optimally



orthogonal representation

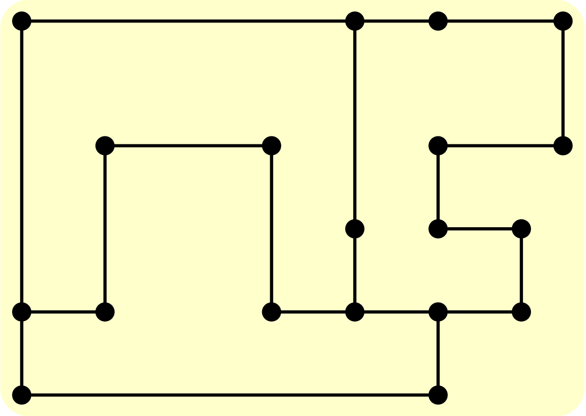
Drawing Turn-Regular Representations Optimally



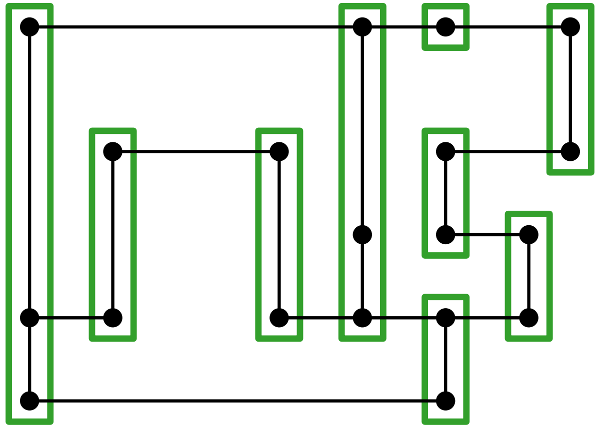


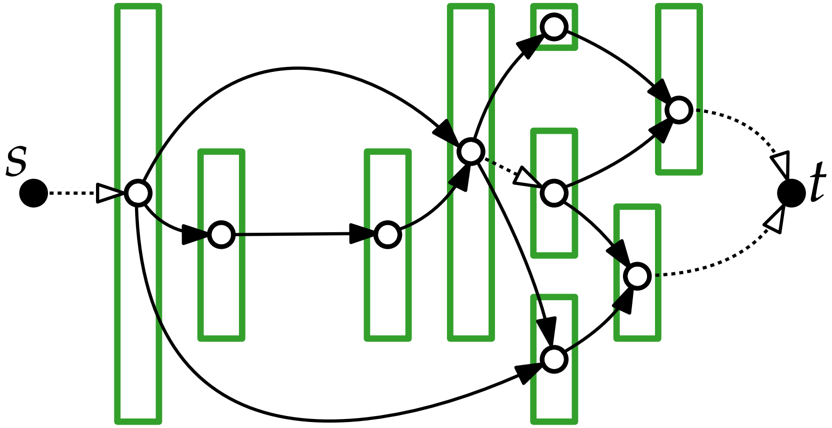
orthogonal representation

Drawing Turn-Regular Representations Optimally



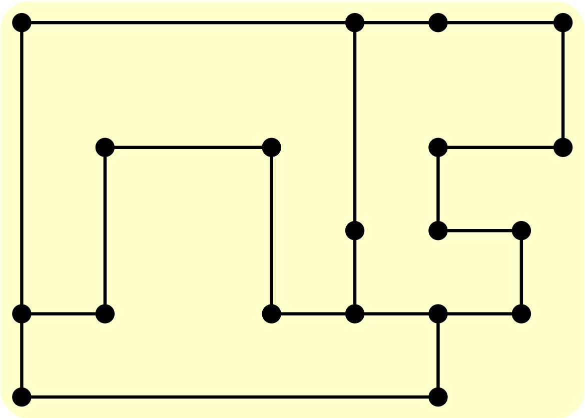
orthogonal representation

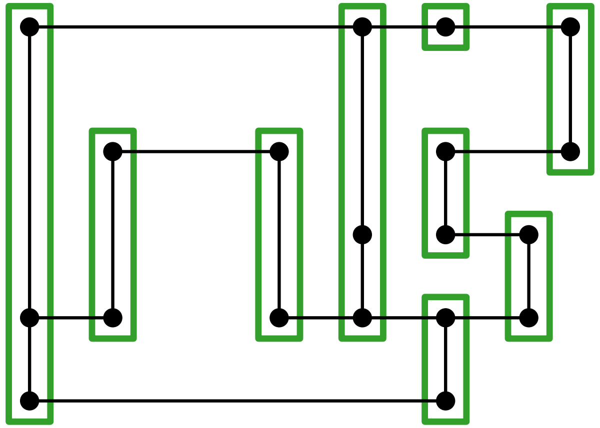


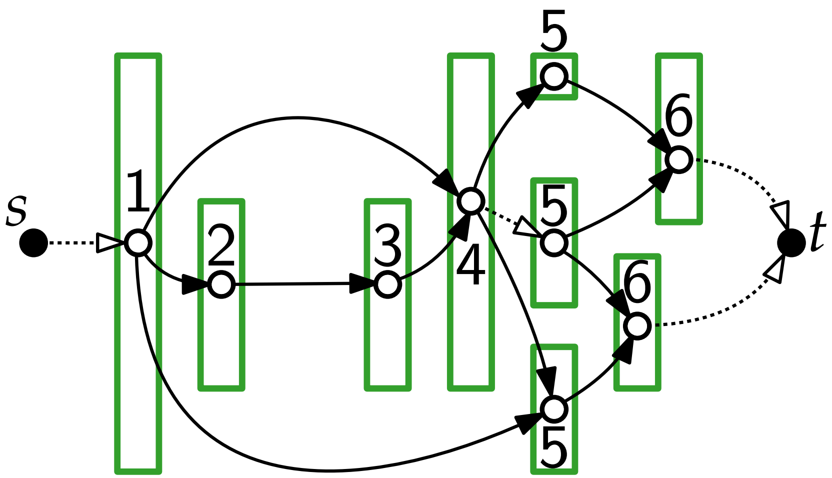


auxiliary graph (DAG) D_x

Drawing Turn-Regular Representations Optimally



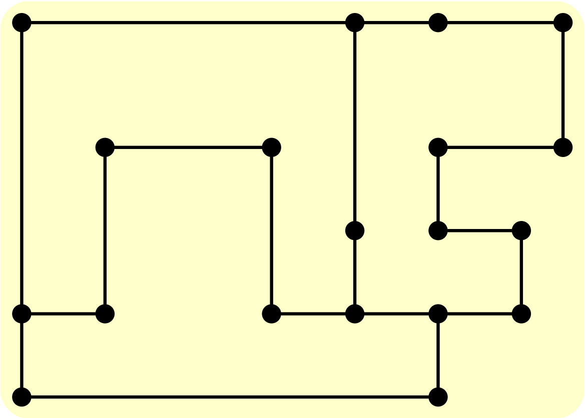




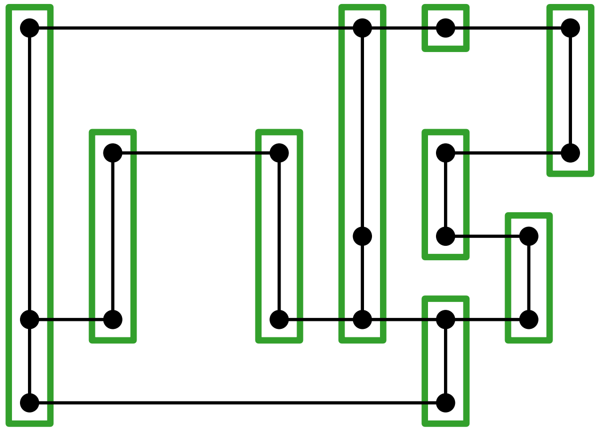
auxiliary graph (DAG) D_x

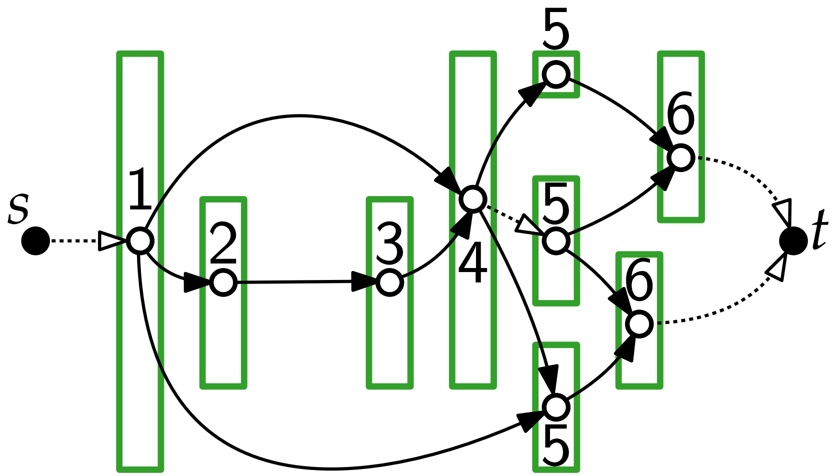
orthogonal representation

Drawing Turn-Regular Representations Optimally

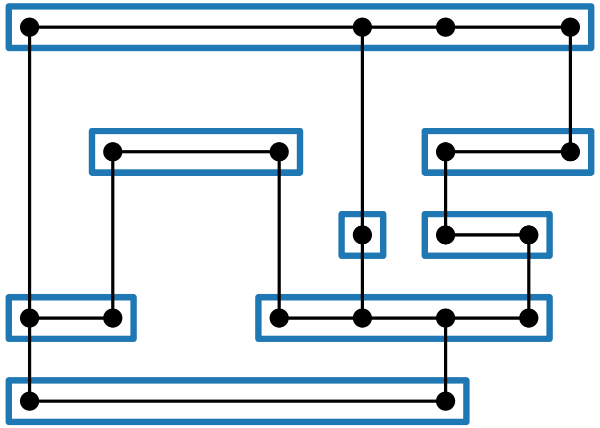


orthogonal representation

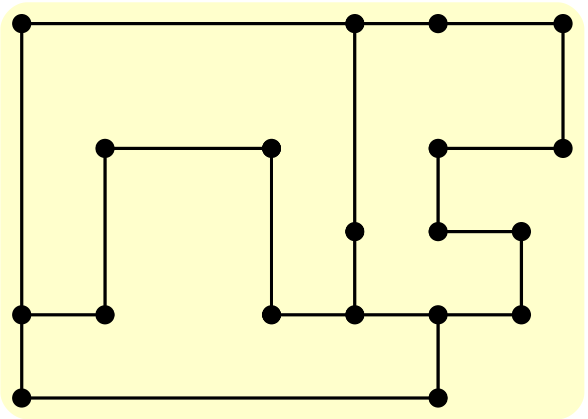




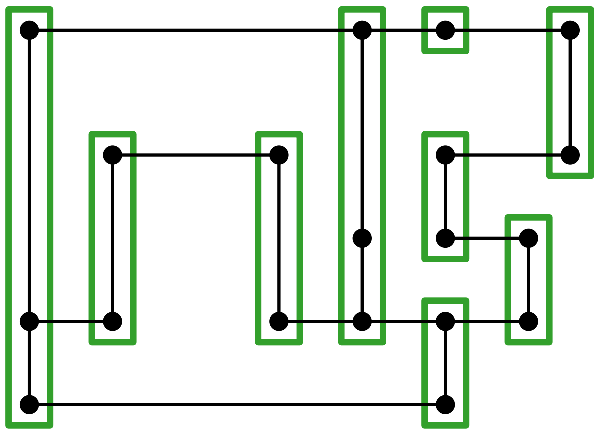
auxiliary graph (DAG) D_x

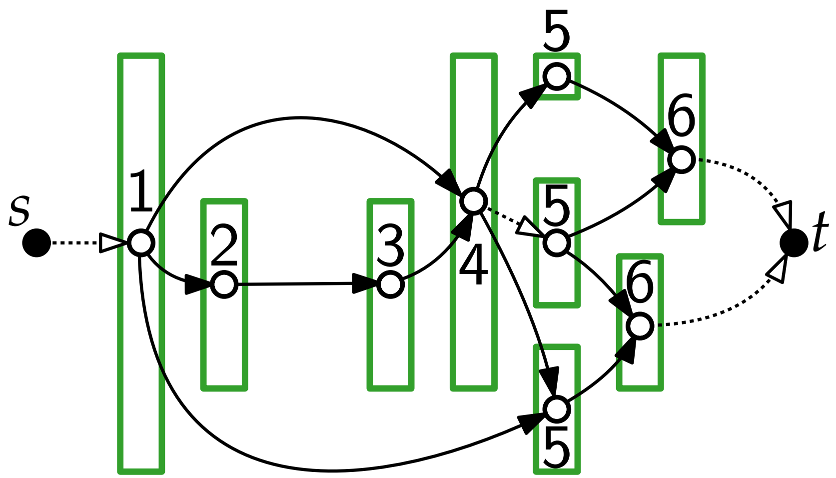


Drawing Turn-Regular Representations Optimally

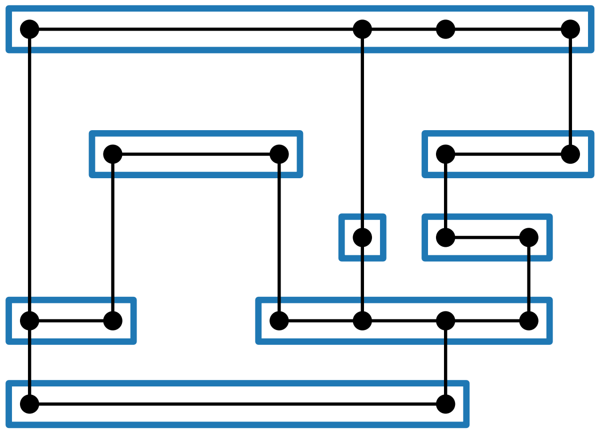


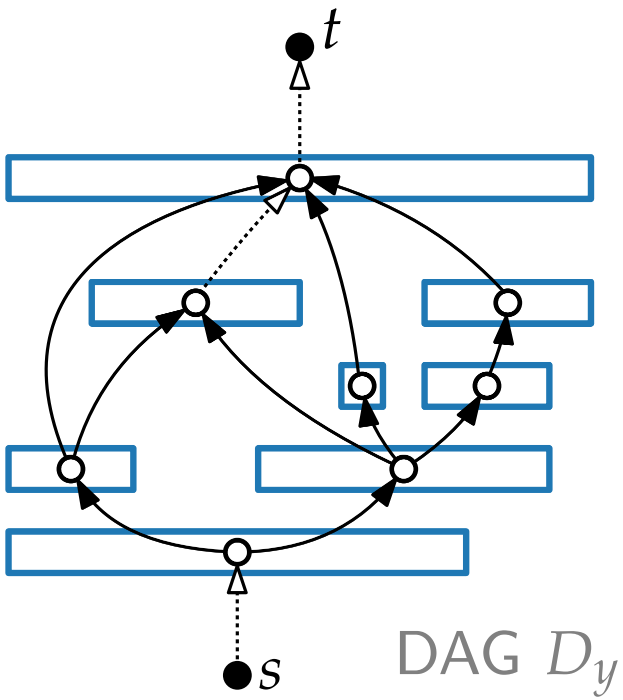
orthogonal representation





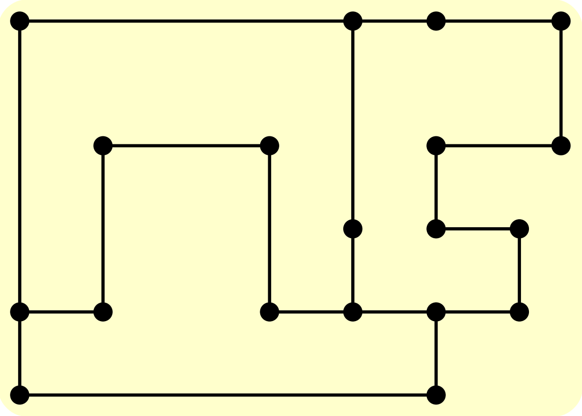
auxiliary graph (DAG) D_x



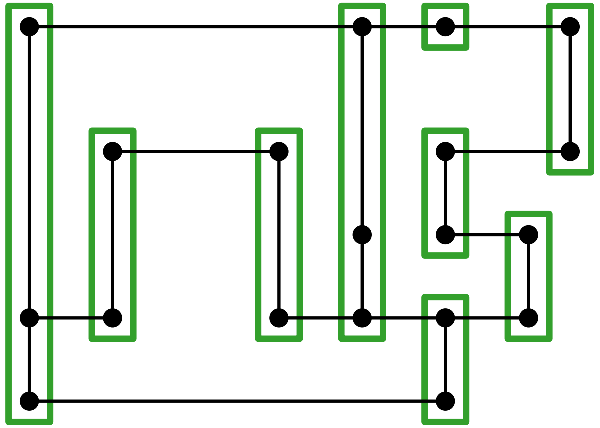


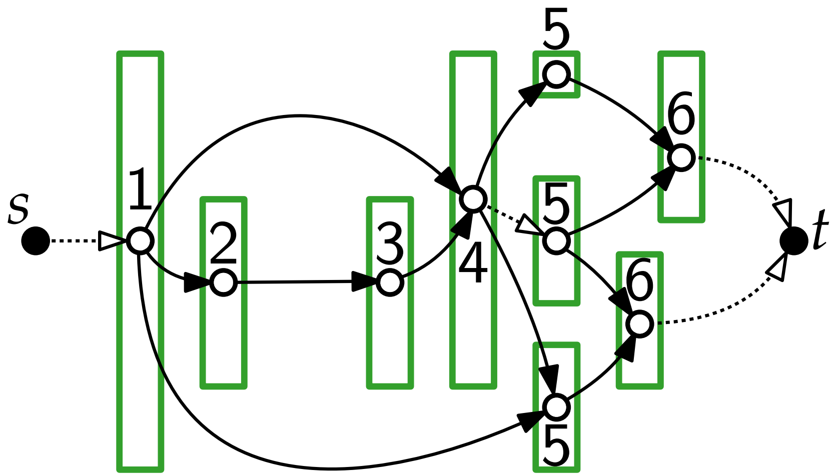
DAG D_y

Drawing Turn-Regular Representations Optimally

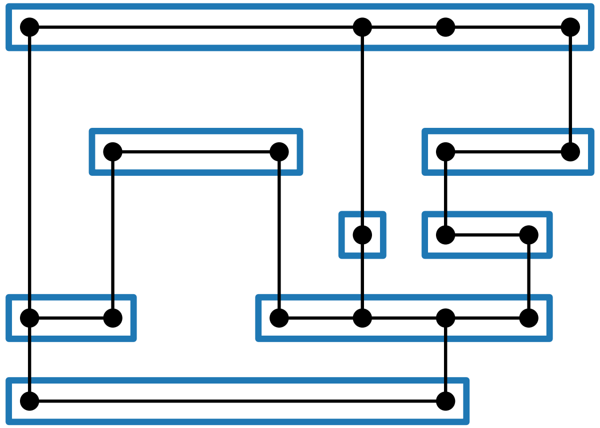


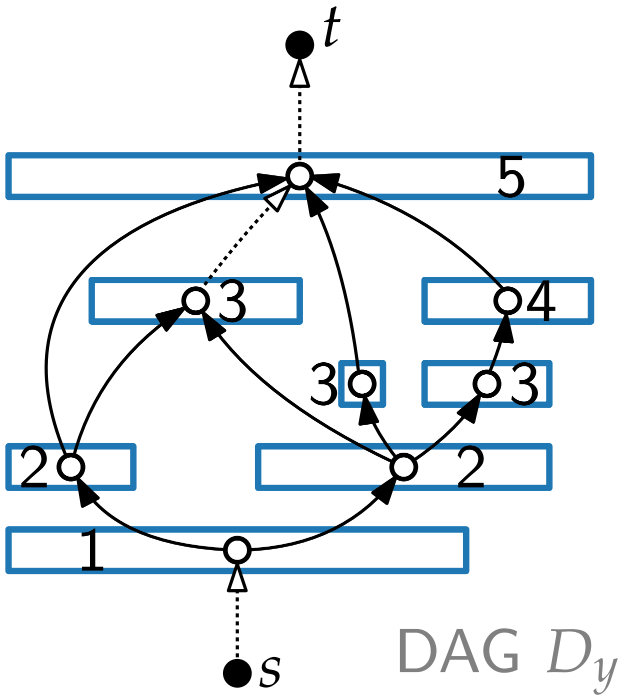
orthogonal representation





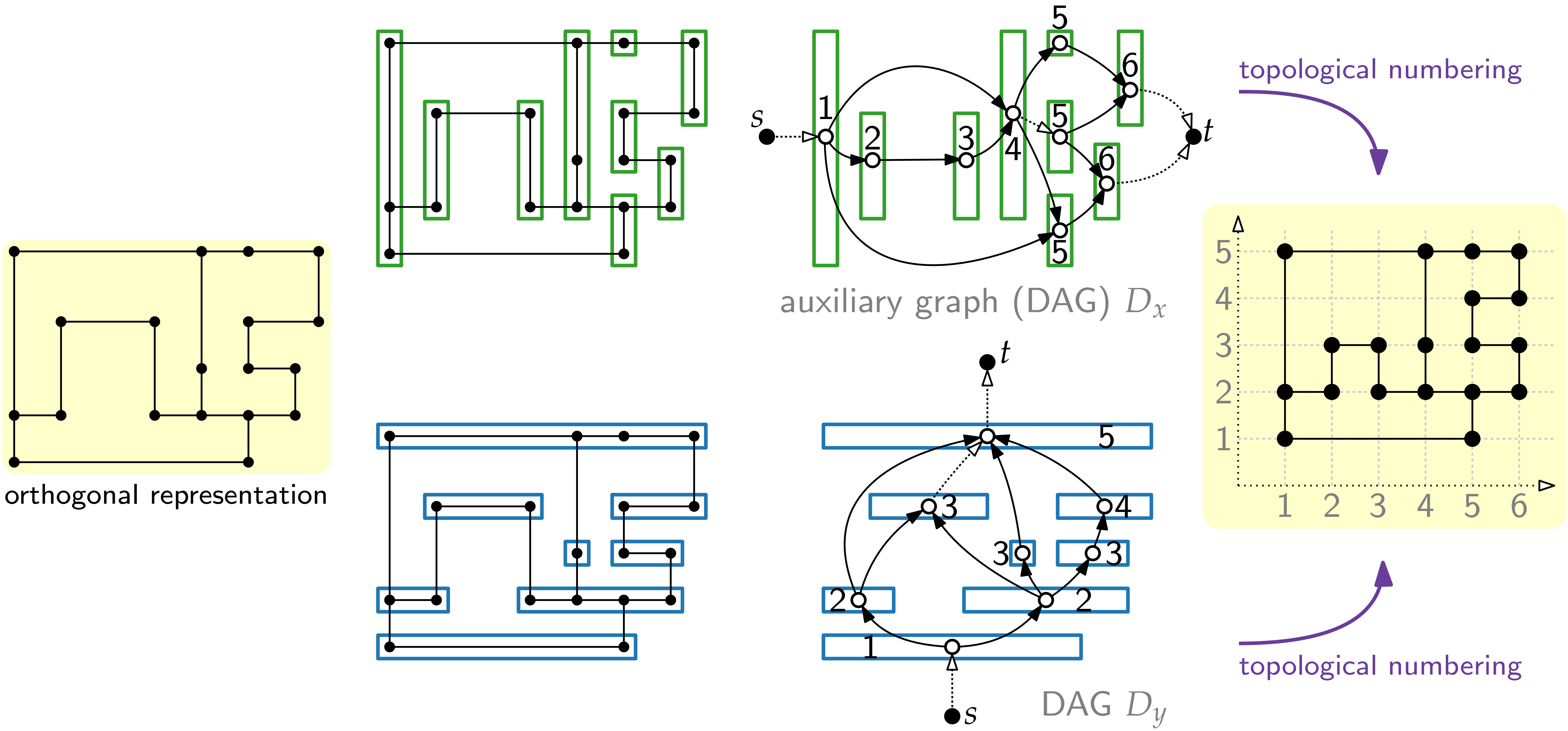
auxiliary graph (DAG) D_x





DAG D_y

Drawing Turn-Regular Representations Optimally



Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

- **Number of faces:**

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.
  For *one* face (cycle), we show the *existence of a polynomial kernel for OC*.
  when parametrized by the number of kitty corners.

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)

  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.

  For *one* face (cycle), we show the *existence of a polynomial kernel for OC*.
  when parametrized by the number of kitty corners.

- **Maximum face-degree:**

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.
  For *one* face (cycle), we show the *existence of a polynomial kernel for OC*.
  when parametrized by the number of kitty corners.

- **Maximum face-degree:** The reductions of Patrignani & Evans et al. use linear-size faces.

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.
  For *one* face (cycle), we show the *existence of a polynomial kernel for OC*.
  when parametrized by the number of kitty corners.

- **Maximum face-degree:** The reductions of Patrignani & Evans et al. use linear-size faces.
  We show: *OC remains NP-hard when parametrized by maximum face degree.*

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.
  For *one* face (cycle), we show the *existence of a polynomial kernel for OC*.
  when parametrized by the number of kitty corners.

- **Maximum face-degree:** The reductions of Patrignani & Evans et al. use linear-size faces.
  We show: *OC remains NP-hard when parametrized by maximum face degree.*

- **Height:**

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.
  For *one* face (cycle), we show the *existence of a polynomial kernel for OC*.
  when parametrized by the number of kitty corners.

- **Maximum face-degree:** The reductions of Patrignani & Evans et al. use linear-size faces.
  We show: *OC remains NP-hard when parametrized by maximum face degree.*

- **Height:** (minimum number of distinct y-coordinates required to draw the representation)

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.
  For *one* face (cycle), we show the *existence of a polynomial kernel for OC*.
  when parametrized by the number of kitty corners.

- **Maximum face-degree:** The reductions of Patrignani & Evans et al. use linear-size faces.
  We show: *OC remains NP-hard when parametrized by maximum face degree.*

- **Height:** (minimum number of distinct y-coordinates required to draw the representation)
  A $(w \times h)$-grid has pathwidth at most h.

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.
  For *one* face (cycle), we show the *existence of a polynomial kernel for OC*.
  when parametrized by the number of kitty corners.

- **Maximum face-degree:** The reductions of Patrignani & Evans et al. use linear-size faces.
  We show: *OC remains NP-hard when parametrized by maximum face degree.*

- **Height:** (minimum number of distinct y-coordinates required to draw the representation)
  A $(w \times h)$-grid has pathwidth at most h.
  $\Rightarrow$ Graphs of bounded height have bounded pathwidth.

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.
  For *one* face (cycle), we show the *existence of a polynomial kernel for OC*.
  when parametrized by the number of kitty corners.

- **Maximum face-degree:** The reductions of Patrignani & Evans et al. use linear-size faces.
  We show: *OC remains NP-hard when parametrized by maximum face degree.*

- **Height:** (minimum number of distinct y-coordinates required to draw the representation)
  A $(w \times h)$-grid has pathwidth at most h.
  $\Rightarrow$ Graphs of bounded height have bounded pathwidth. But converse generally not true.

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.
  For *one* face (cycle), we show the *existence of a polynomial kernel for OC*.
  when parametrized by the number of kitty corners.

- **Maximum face-degree:** The reductions of Patrignani & Evans et al. use linear-size faces.
  We show: *OC remains NP-hard when parametrized by maximum face degree.*

- **Height:** (minimum number of distinct y-coordinates required to draw the representation)
  A $(w \times h)$-grid has pathwidth at most h.
  $\Rightarrow$ Graphs of bounded height have bounded pathwidth. But converse generally not true.
  We show: *OC admits an XP algorithm parametrized by height.*

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.
  For *one* face (cycle), we show the *existence of a polynomial kernel for OC*.
  when parametrized by the number of kitty corners.

- **Maximum face-degree:** The reductions of Patrignani & Evans et al. use linear-size faces.
  We show: *OC remains NP-hard when parametrized by maximum face degree.*

- **Height:** (minimum number of distinct y-coordinates required to draw the representation)
  A $(w \times h)$-grid has pathwidth at most h.
  $\Rightarrow$ Graphs of bounded height have bounded pathwidth. But converse generally not true.
  We show: *OC admits an XP algorithm parametrized by height.* (Just a DP, left to right.)

Our Contribution: A Parametrized View of OC

- **Number of kitty corners:** (the number of vertices involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

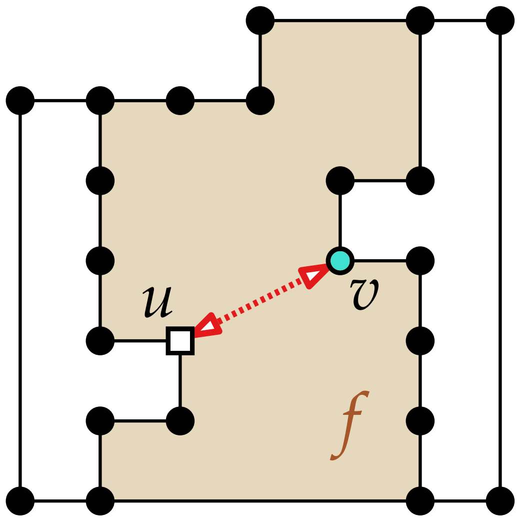
- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.
  For *one* face (cycle), we show the *existence of a polynomial kernel for OC*.
  when parametrized by the number of kitty corners.

- **Maximum face-degree:** The reductions of Patrignani & Evans et al. use linear-size faces.
  We show: *OC remains NP-hard when parametrized by maximum face degree.*

- **Height:** (minimum number of distinct y-coordinates required to draw the representation)
  A $(w \times h)$-grid has pathwidth at most h.
  $\Rightarrow$ Graphs of bounded height have bounded pathwidth. But converse generally not true.
  We show: *OC admits an XP algorithm parametrized by height.* (Just a DP, left to right.)

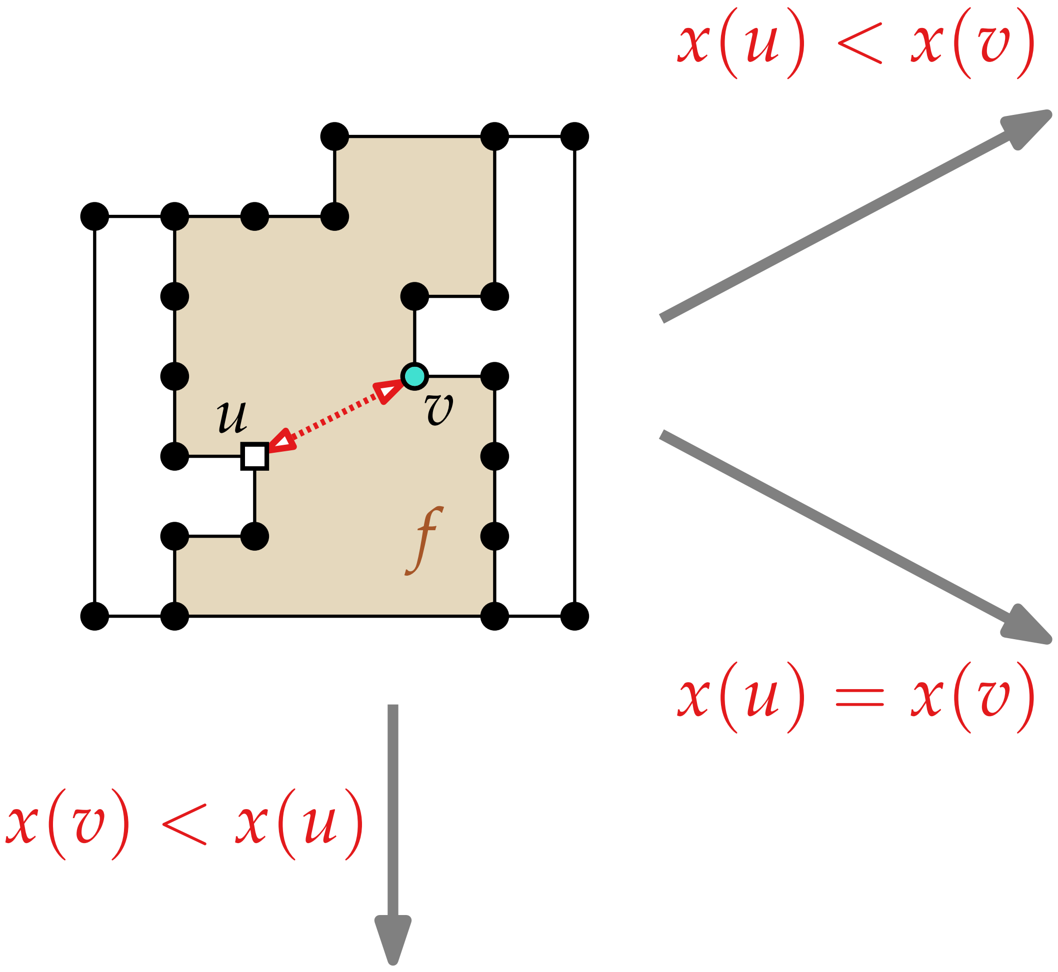
An FPT Algorithm for OC, Parametrized by # Kitty Corners

Idea: For each (?) pair $\{u, v\}$ of kitty corners, guess whether $x(u) \lesseqqgtr x(v)$ and $y(u) \lesseqqgtr y(v)$.



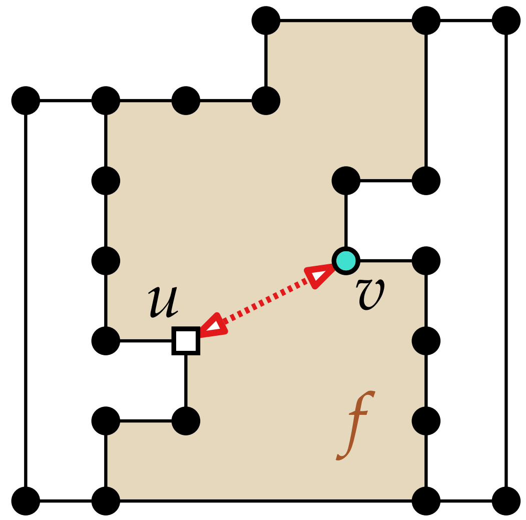
An FPT Algorithm for OC, Parametrized by # Kitty Corners

Idea: For each (?) pair $\{u, v\}$ of kitty corners, guess whether $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$.

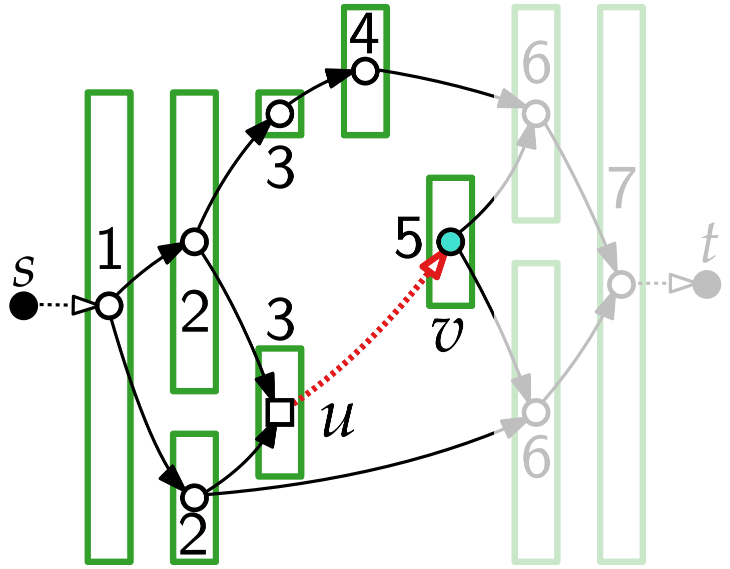


An FPT Algorithm for OC, Parametrized by # Kitty Corners

Idea: For each (?) pair $\{u, v\}$ of kitty corners, guess whether $x(u) \lesseqqgtr x(v)$ and $y(u) \lesseqqgtr y(v)$.



$x(u) < x(v)$

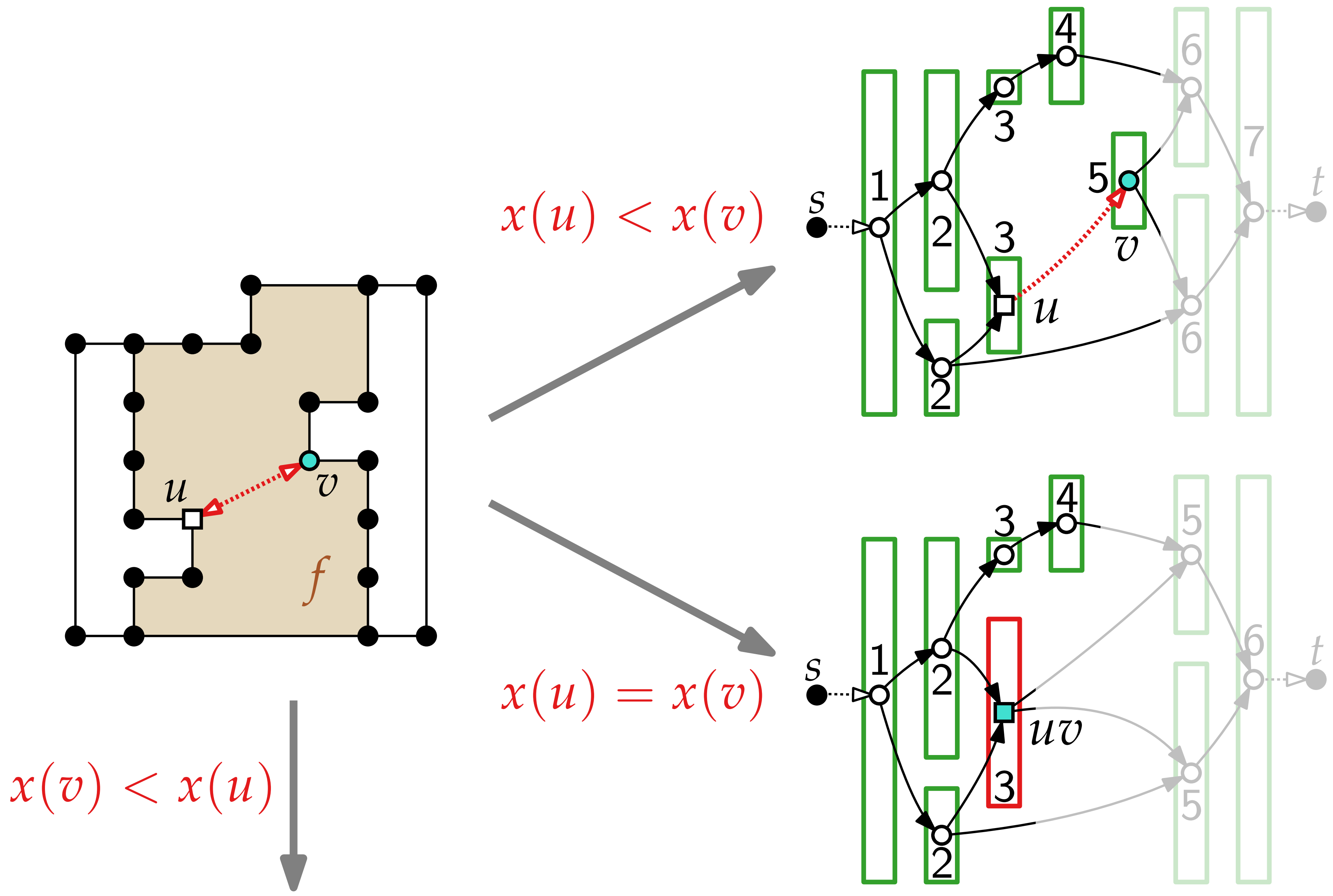


$x(u) = x(v)$

$x(v) < x(u)$

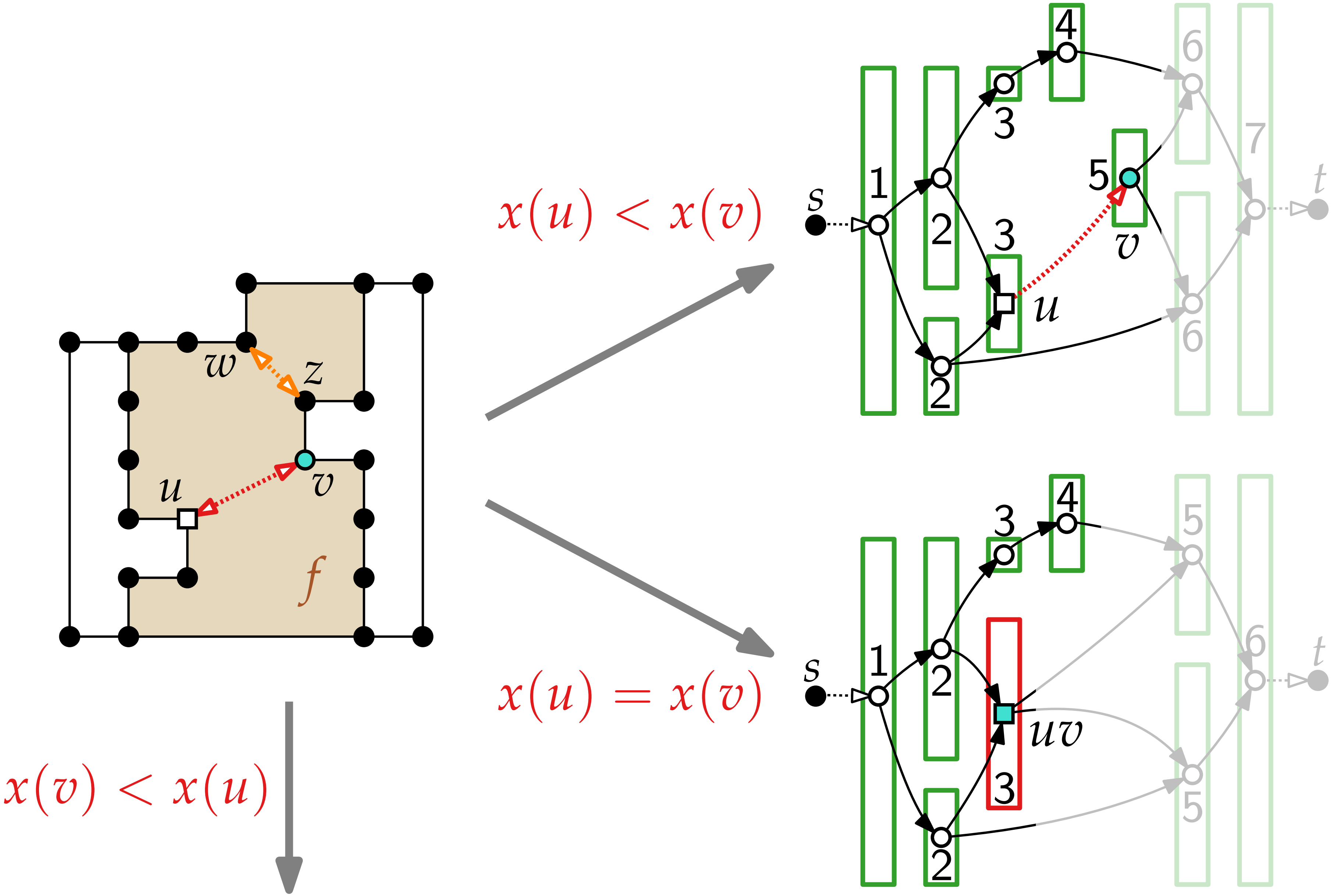
An FPT Algorithm for OC, Parametrized by # Kitty Corners

Idea: For each (?) pair $\{u, v\}$ of kitty corners, guess whether $x(u) \overset{<}{\underset{>}{=}} x(v)$ and $y(u) \overset{<}{\underset{>}{=}} y(v)$.



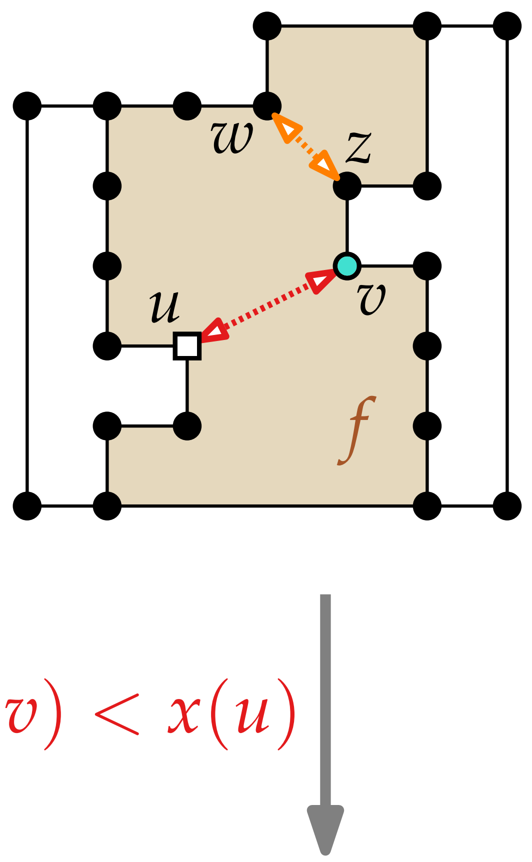
An FPT Algorithm for OC, Parametrized by # Kitty Corners

Idea: For each (?) pair $\{u, v\}$ of kitty corners, guess whether $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$.



An FPT Algorithm for OC, Parametrized by # Kitty Corners

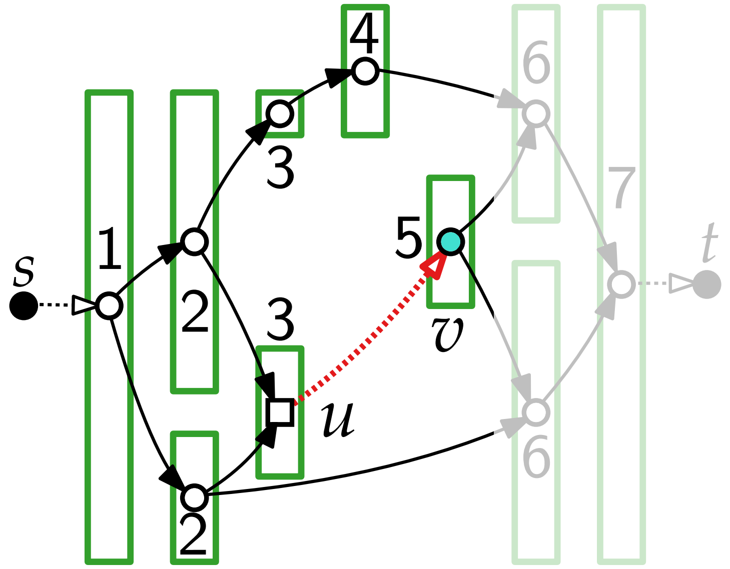
Idea: For each (?) pair $\{u, v\}$ of kitty corners, guess whether $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$.



$x(v) < x(u)$

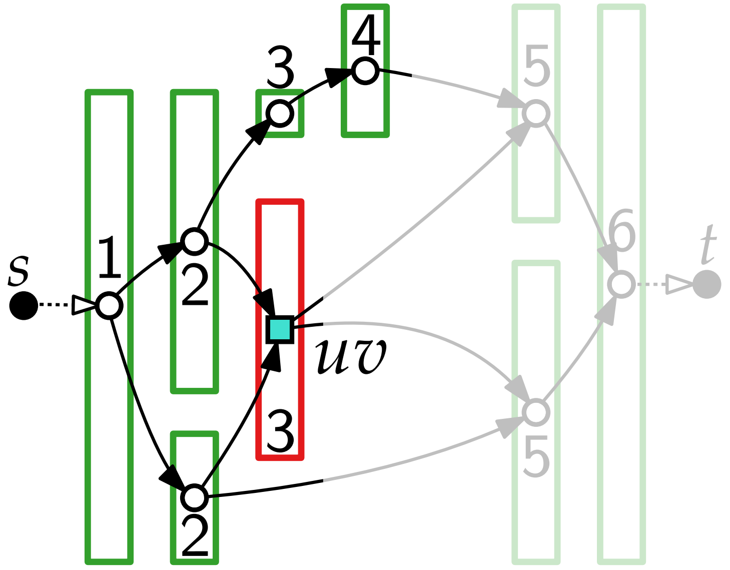
$x(w) < x(z)$

$x(u) < x(v)$



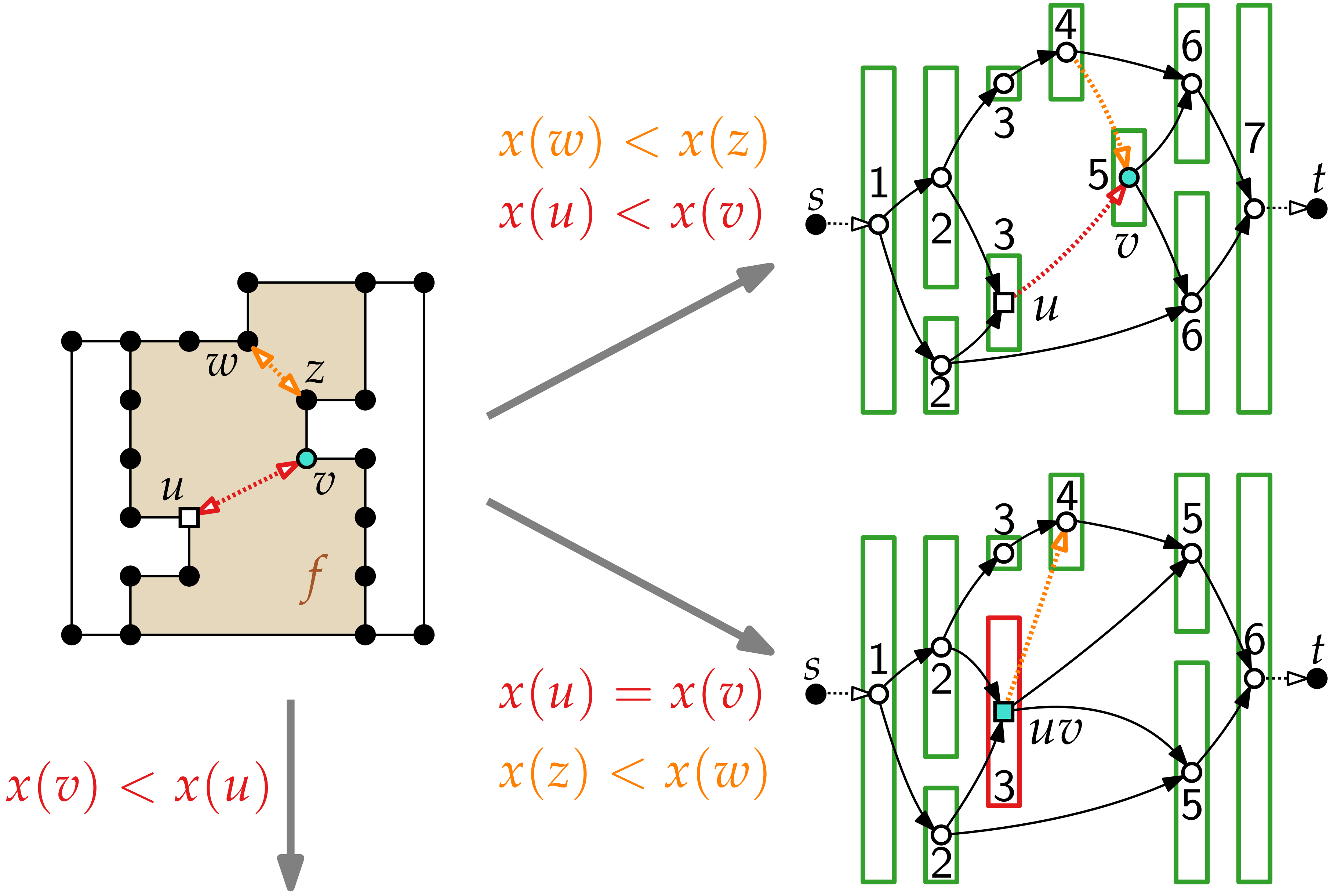
$x(u) = x(v)$

$x(z) < x(w)$



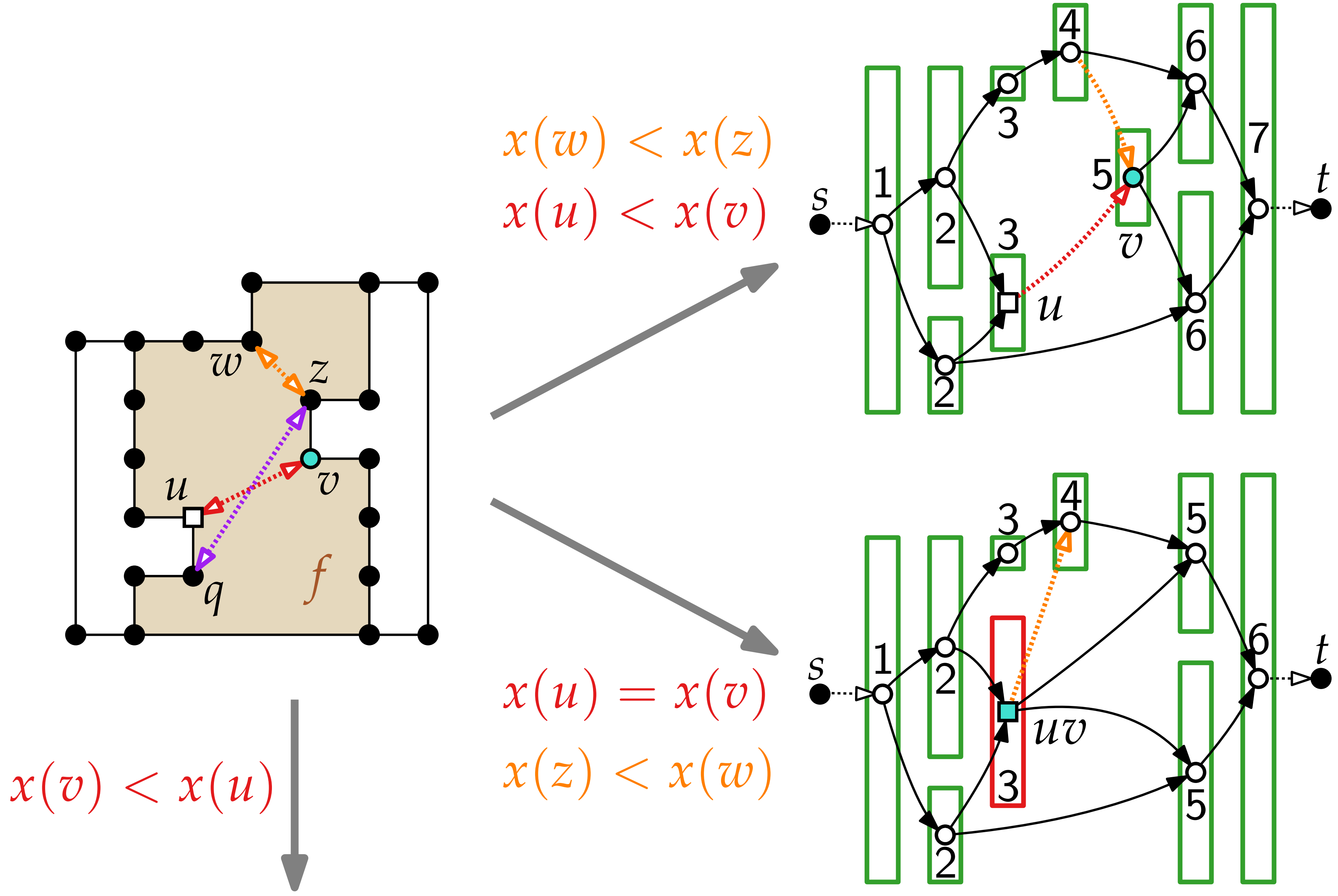
An FPT Algorithm for OC, Parametrized by # Kitty Corners

Idea: For each (?) pair $\{u, v\}$ of kitty corners, guess whether $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$.



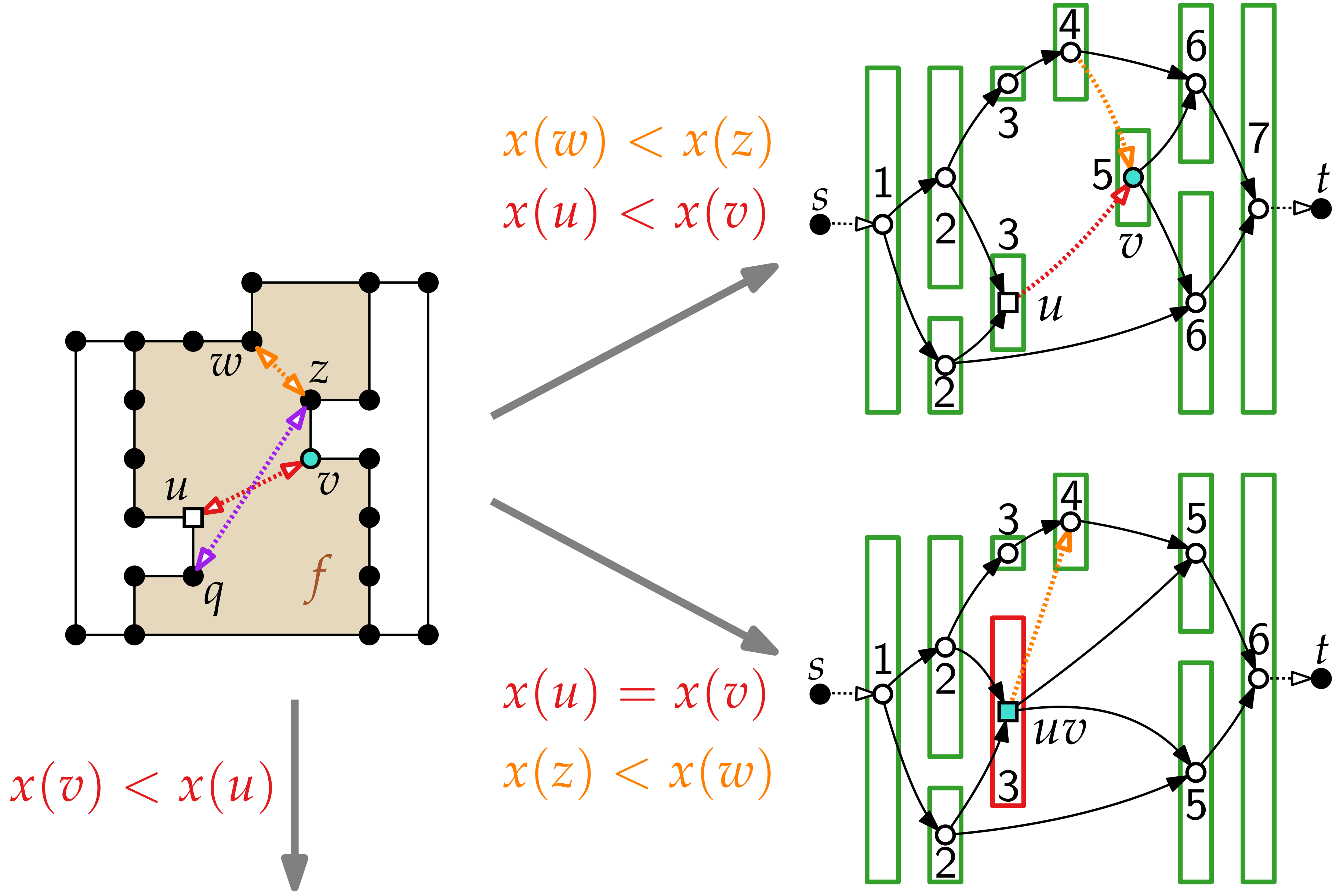
An FPT Algorithm for OC, Parametrized by # Kitty Corners

Idea: For each (?) pair $\{u, v\}$ of kitty corners, guess whether $x(u) \substack{<\\=\\>} x(v)$ and $y(u) \substack{<\\=\\>} y(v)$.



An FPT Algorithm for OC, Parametrized by # Kitty Corners

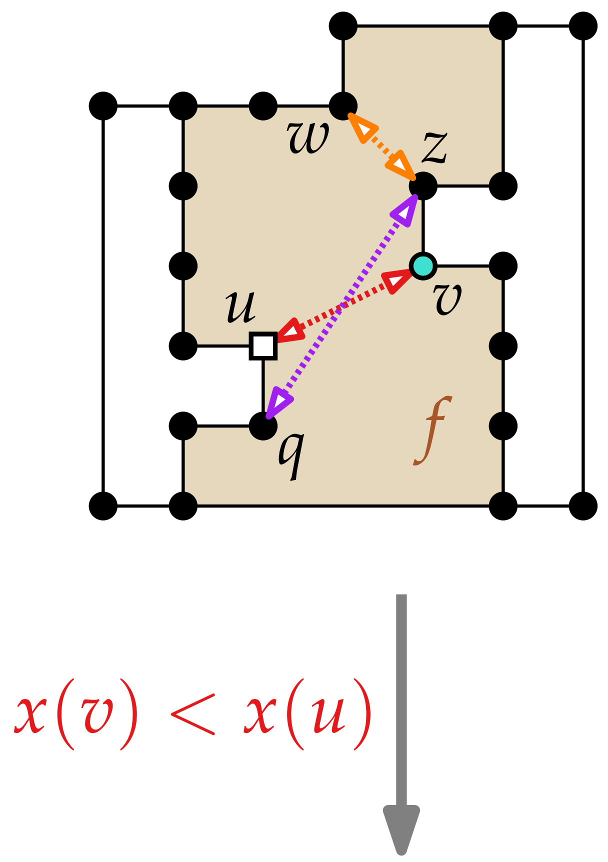
Idea: For each (?) pair $\{u, v\}$ of kitty corners, guess whether $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$.

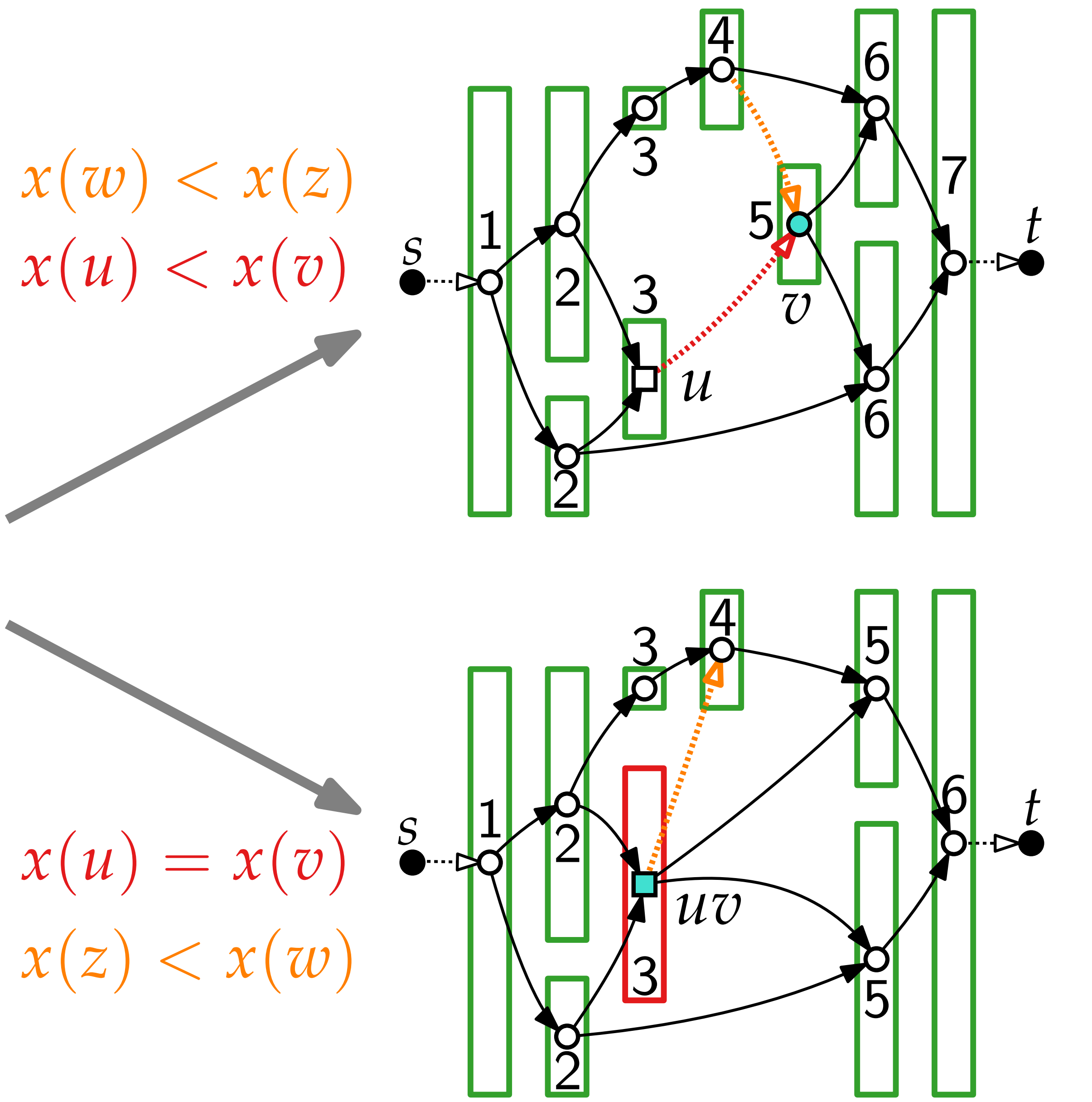


Question:
Do we need to deal with $\{q, z\}$ analogously?

An FPT Algorithm for OC, Parametrized by # Kitty Corners

Idea: For each (?) pair $\{u, v\}$ of kitty corners, guess whether $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$.



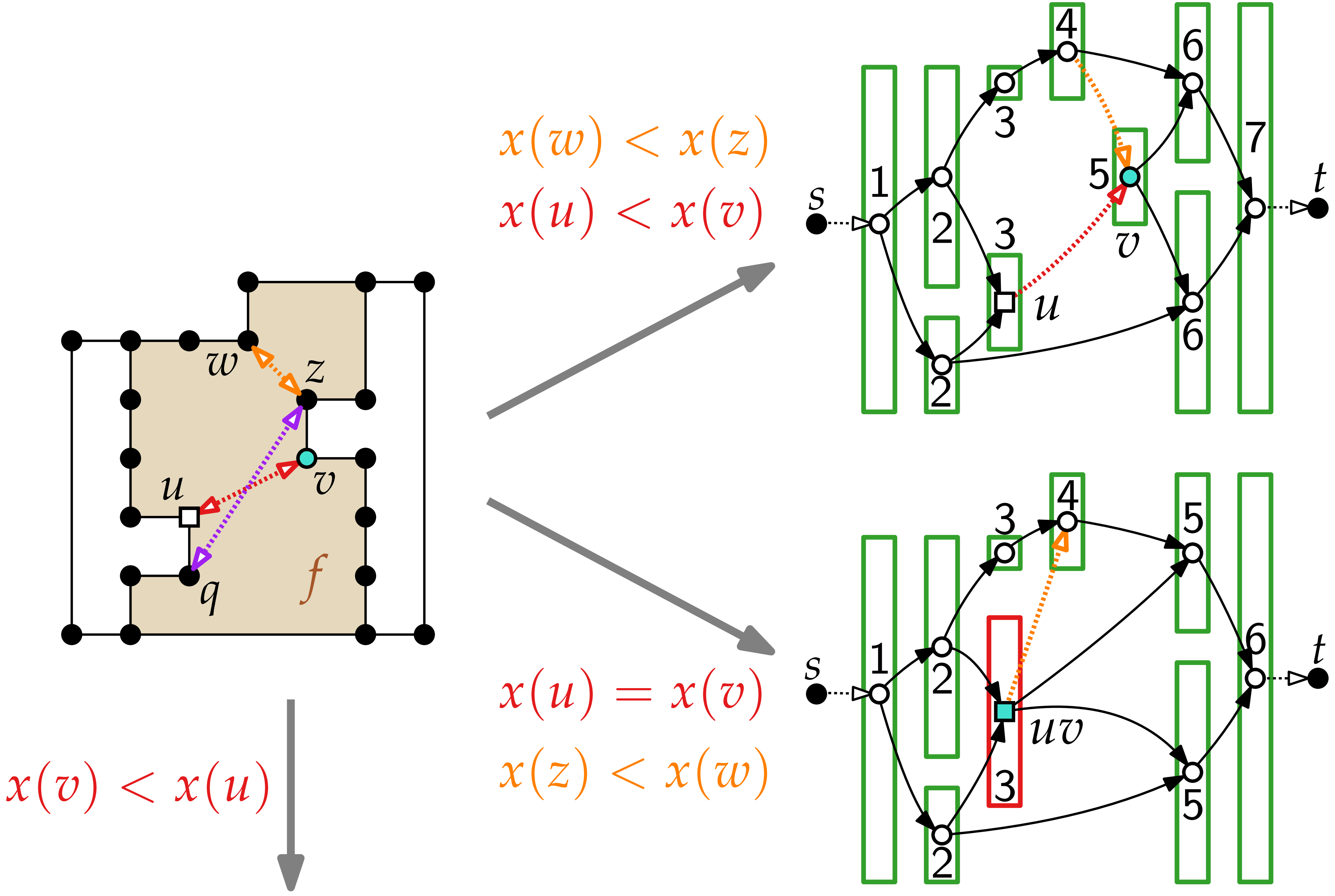


Question:
Do we need to deal with $\{q, z\}$ analogously?

Answer:
No, doesn't add new information.

An FPT Algorithm for OC, Parametrized by # Kitty Corners

Idea: For each (?) pair $\{u, v\}$ of kitty corners, guess whether $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$.



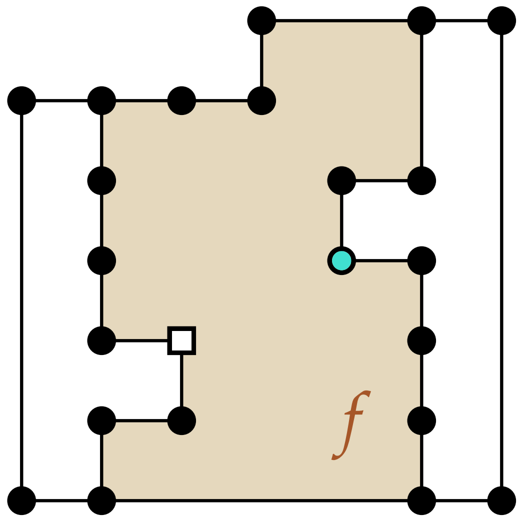
Question:
Do we need to deal with $\{q, z\}$ analogously?

Answer:
No, doesn't add new information.

Thus:
Suffices to consider a *maximal planar subset* of the set K_f of "kitty corner edges" of f.

An FPT Algorithm for OC, cont'd

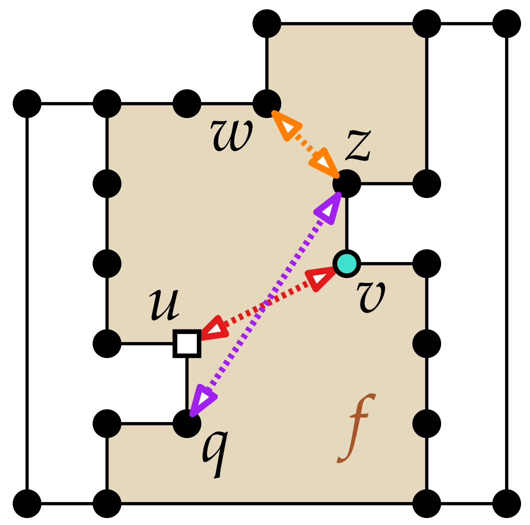
Let f be a face of the given orthogonal representation of G.



An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

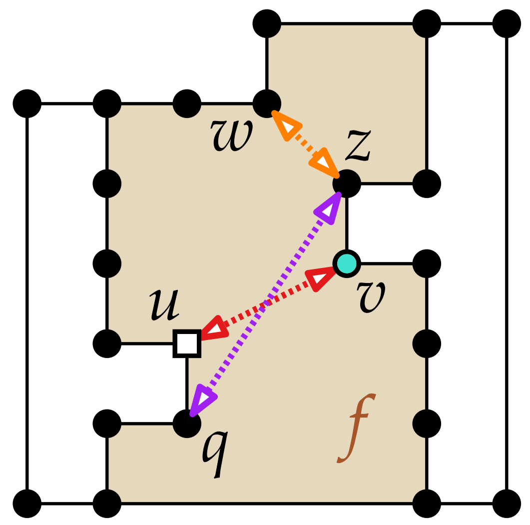
- Let K_f be the set of kitty corner edges in f.



An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

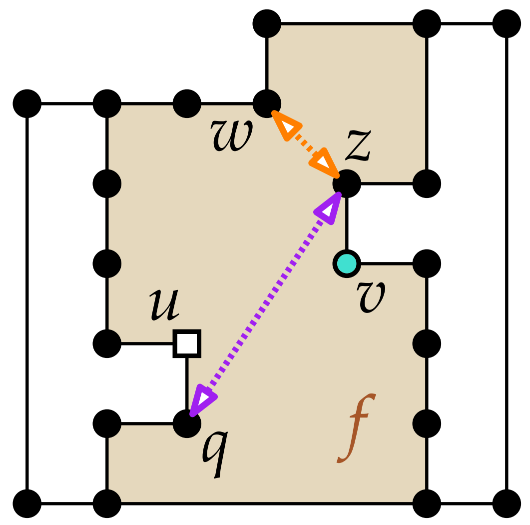
- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.



An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

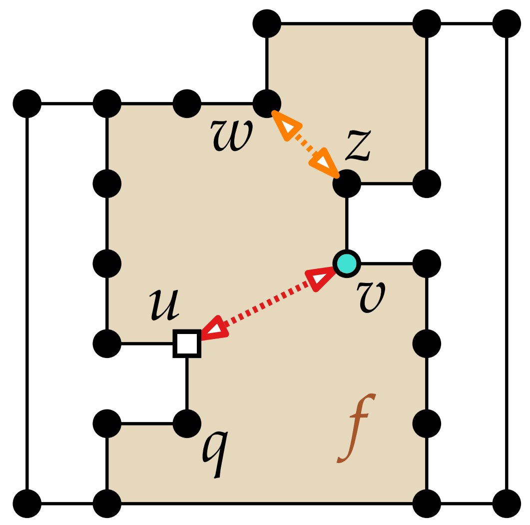
- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.



An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

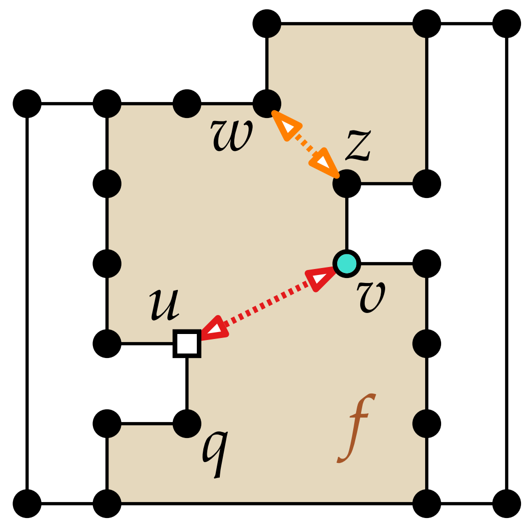
- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.



An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

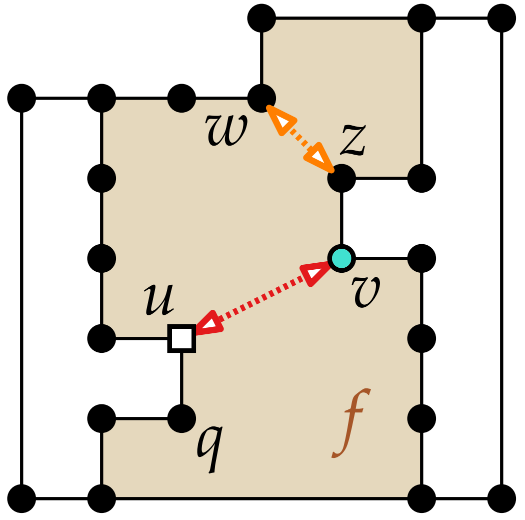
- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \ldots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.



An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

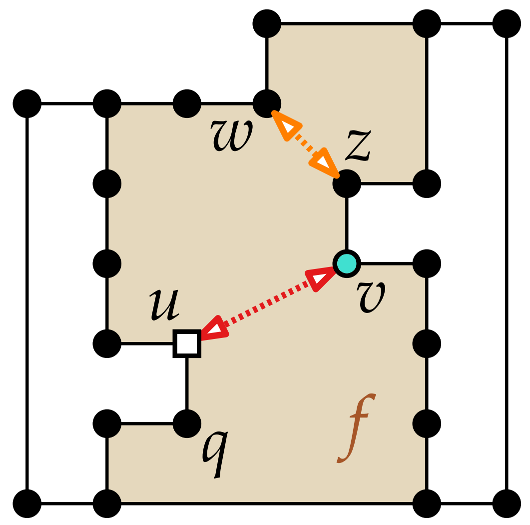
- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \ldots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.
  - For each edge $\{u, v\}$ in E_p, try adding each of $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$ to D_x / D_y.



An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

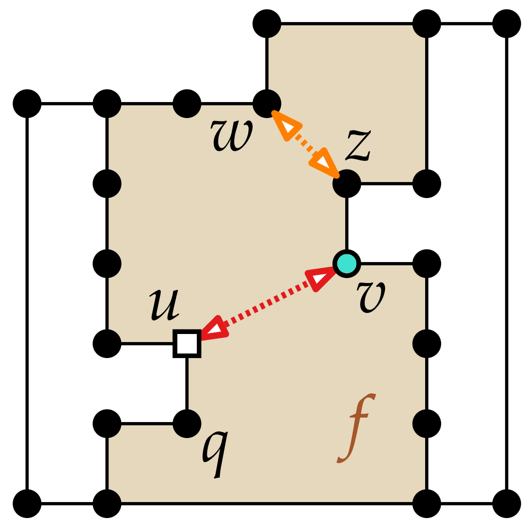
- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \ldots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.
  - For each edge $\{u, v\}$ in E_p, try adding each of $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$ to D_x / D_y.
  - Let $\mathcal{D}_x / \mathcal{D}_y$ be the resulting sets of DAGs. For each combination $(D_x, D_y) \in \mathcal{D}_x \times \mathcal{D}_y$:



An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

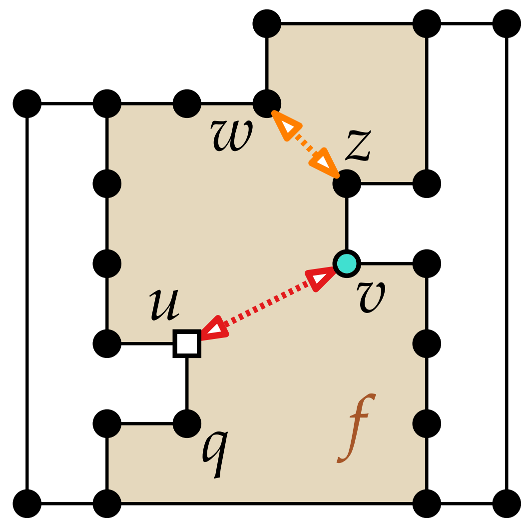
- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \dots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.
  - For each edge $\{u, v\}$ in E_p, try adding each of $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$ to D_x / D_y.
  - Let $\mathcal{D}_x / \mathcal{D}_y$ be the resulting sets of DAGs. For each combination $(D_x, D_y) \in \mathcal{D}_x \times \mathcal{D}_y$:
    - Run the algorithm for the turn-regular case.



An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

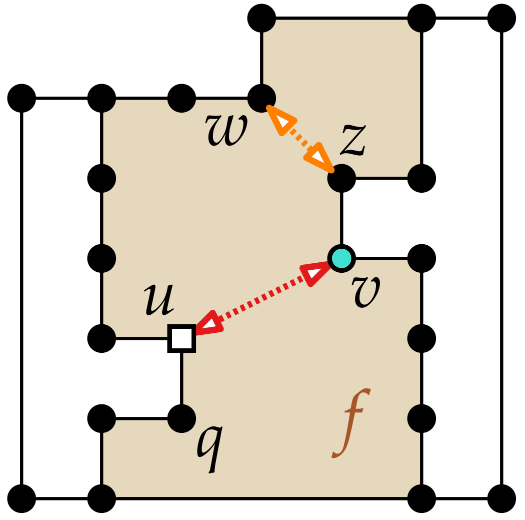
- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \ldots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.
  - For each edge $\{u, v\}$ in E_p, try adding each of $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$ to D_x / D_y.
  - Let $\mathcal{D}_x / \mathcal{D}_y$ be the resulting sets of DAGs. For each combination $(D_x, D_y) \in \mathcal{D}_x \times \mathcal{D}_y$:
    - Run the algorithm for the turn-regular case.
    - If the resulting drawing does not self-intersect, measure its area.



An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

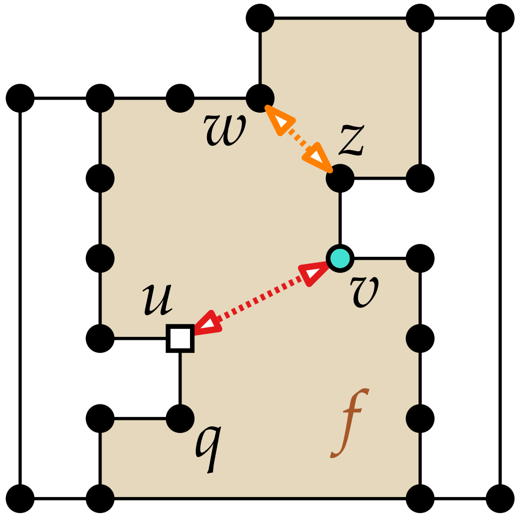
- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \dots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.
  - For each edge $\{u, v\}$ in E_p, try adding each of $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$ to D_x / D_y.
  - Let $\mathcal{D}_x / \mathcal{D}_y$ be the resulting sets of DAGs. For each combination $(D_x, D_y) \in \mathcal{D}_x \times \mathcal{D}_y$:
    - Run the algorithm for the turn-regular case.
    - If the resulting drawing does not self-intersect, measure its area.
- Return the drawing of minimum area.



An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \ldots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.
  - For each edge $\{u, v\}$ in E_p, try adding each of $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$ to D_x / D_y.
  - Let $\mathcal{D}_x / \mathcal{D}_y$ be the resulting sets of DAGs. For each combination $(D_x, D_y) \in \mathcal{D}_x \times \mathcal{D}_y$:
    - Run the algorithm for the turn-regular case.
    - If the resulting drawing does not self-intersect, measure its area.
- Return the drawing of minimum area.

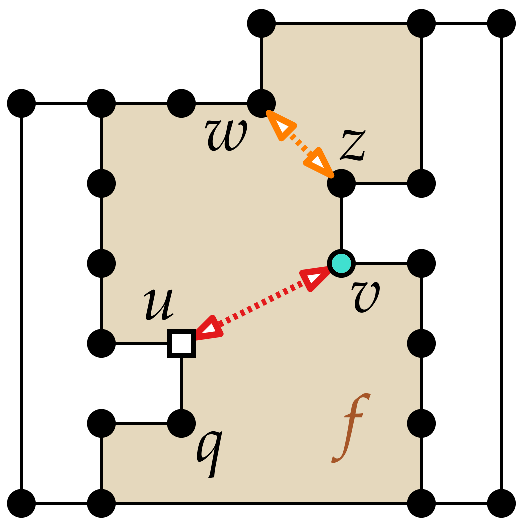


Runtime: Let $k_f = \#$kitty corners in face f.

An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \ldots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.
  - For each edge $\{u, v\}$ in E_p, try adding each of $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$ to D_x / D_y.
  - Let $\mathcal{D}_x / \mathcal{D}_y$ be the resulting sets of DAGs. For each combination $(D_x, D_y) \in \mathcal{D}_x \times \mathcal{D}_y$:
    - Run the algorithm for the turn-regular case.
    - If the resulting drawing does not self-intersect, measure its area.
- Return the drawing of minimum area.



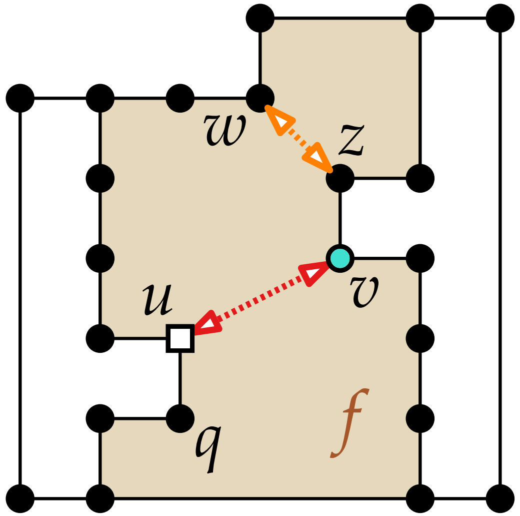
Runtime: Let $k_f = \#$kitty corners in face f.
$\#$ maximal planar subsets of K_f $=$

An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \dots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.
  - For each edge $\{u, v\}$ in E_p, try adding each of $x(u) \lessgtr x(v)$ and $y(u) \lessgtr y(v)$ to D_x / D_y.
  - Let $\mathcal{D}_x / \mathcal{D}_y$ be the resulting sets of DAGs. For each combination $(D_x, D_y) \in \mathcal{D}_x \times \mathcal{D}_y$:
    - Run the algorithm for the turn-regular case.
    - If the resulting drawing does not self-intersect, measure its area.
- Return the drawing of minimum area.

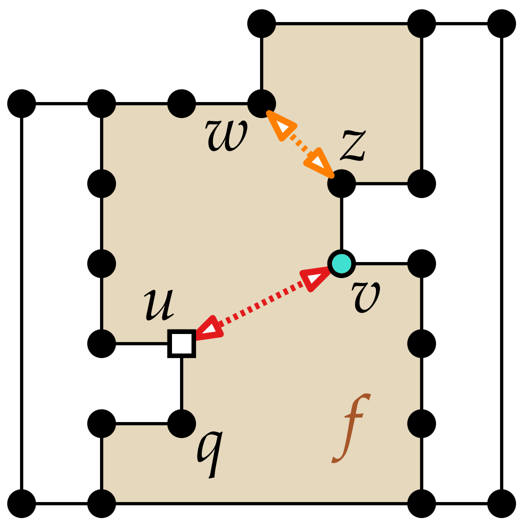
Runtime: Let $k_f = \#$kitty corners in face f.
$\#$maximal planar subsets of $K_f = \#$outerplanar graphs on k_f vertices



An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \ldots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.
  - For each edge $\{u, v\}$ in E_p, try adding each of $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$ to D_x / D_y.
  - Let $\mathcal{D}_x / \mathcal{D}_y$ be the resulting sets of DAGs. For each combination $(D_x, D_y) \in \mathcal{D}_x \times \mathcal{D}_y$:
    - Run the algorithm for the turn-regular case.
    - If the resulting drawing does not self-intersect, measure its area.
- Return the drawing of minimum area.

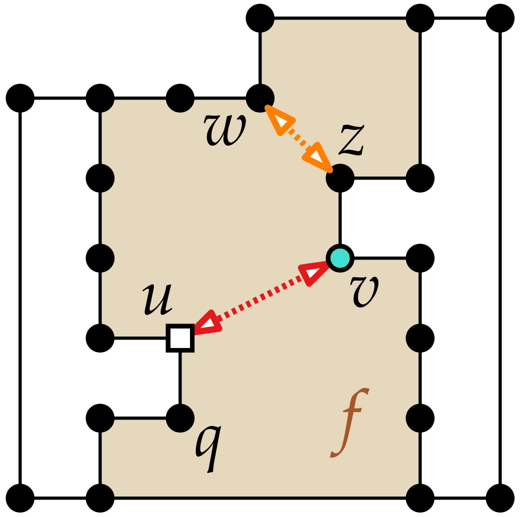


Runtime: Let $k_f = \#$kitty corners in face f.
$\#$maximal planar subsets of $K_f = \#$outerplanar graphs on k_f vertices
$\leq 2^{2k_f - 3}$

An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \dots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.
  - For each edge $\{u, v\}$ in E_p, try adding each of $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$ to D_x / D_y.
  - Let $\mathcal{D}_x / \mathcal{D}_y$ be the resulting sets of DAGs. For each combination $(D_x, D_y) \in \mathcal{D}_x \times \mathcal{D}_y$:
    - Run the algorithm for the turn-regular case.
    - If the resulting drawing does not self-intersect, measure its area.
- Return the drawing of minimum area.

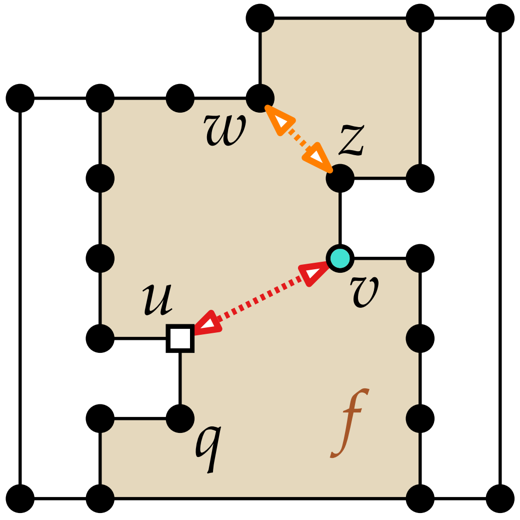


Runtime: Let $k_f = \#$kitty corners in face f.
$\#$maximal planar subsets of $K_f = \#$outerplanar graphs on k_f vertices
$\leq 2^{2k_f - 3} < 4^{k_f}$

An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \dots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.
  - For each edge $\{u, v\}$ in E_p, try adding each of $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$ to D_x / D_y.
  - Let $\mathcal{D}_x / \mathcal{D}_y$ be the resulting sets of DAGs. For each combination $(D_x, D_y) \in \mathcal{D}_x \times \mathcal{D}_y$:
    - Run the algorithm for the turn-regular case.
    - If the resulting drawing does not self-intersect, measure its area.
- Return the drawing of minimum area.

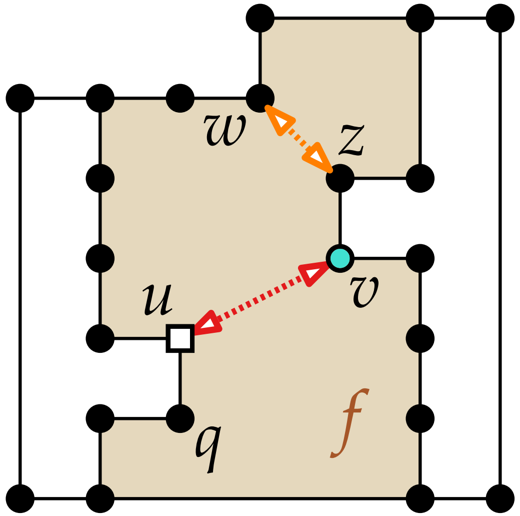


Runtime: Let $k_f = \#$kitty corners in face f.
$\#$maximal planar subsets of $K_f = \#$outerplanar graphs on k_f vertices
$\leq 2^{2k_f - 3} < 4^{k_f} \quad \Rightarrow \#$augmentations for the whole graph: $\Pi_f 4^{k_f}$

An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \ldots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.
  - For each edge $\{u, v\}$ in E_p, try adding each of $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$ to D_x / D_y.
  - Let $\mathcal{D}_x / \mathcal{D}_y$ be the resulting sets of DAGs. For each combination $(D_x, D_y) \in \mathcal{D}_x \times \mathcal{D}_y$:
    - Run the algorithm for the turn-regular case.
    - If the resulting drawing does not self-intersect, measure its area.
- Return the drawing of minimum area.

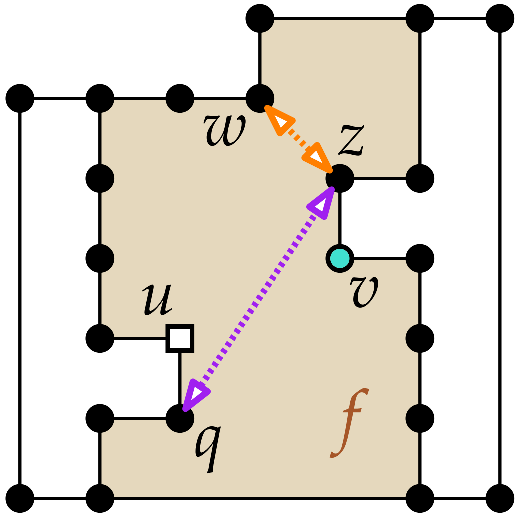


Runtime: Let $k_f = \#$ kitty corners in face f.
$\#$ maximal planar subsets of $K_f = \#$ outerplanar graphs on k_f vertices
$\leq 2^{2k_f - 3} < 4^{k_f} \quad \Rightarrow \#$ augmentations for the whole graph: $\Pi_f 4^{k_f} \leq 4^k$.

An FPT Algorithm for OC, cont'd

Let f be a face of the given orthogonal representation of G.

- Let K_f be the set of kitty corner edges in f.
- Let $\mathcal{P}_f$ be the set of maximal planar subsets of K_f. Let $\mathcal{P} = \Pi_f \mathcal{P}_f$.
- For each element $p = (p_1, \ldots, p_F)$ of $\mathcal{P}$, let E_p be the corresponding set of edges.
  - For each edge $\{u, v\}$ in E_p, try adding each of $x(u) \lesseqgtr x(v)$ and $y(u) \lesseqgtr y(v)$ to D_x / D_y.
  - Let $\mathcal{D}_x / \mathcal{D}_y$ be the resulting sets of DAGs. For each combination $(D_x, D_y) \in \mathcal{D}_x \times \mathcal{D}_y$:
    - Run the algorithm for the turn-regular case.
    - If the resulting drawing does not self-intersect, measure its area.
- Return the drawing of minimum area.



Runtime: Let $k_f = \#$ kitty corners in face f.
$\#$ maximal planar subsets of $K_f = \#$ outerplanar graphs on k_f vertices
$\leq 2^{2k_f - 3} < 4^{k_f} \quad \Rightarrow \#$ augmentations for the whole graph: $\Pi_f 4^{k_f} \leq 4^k$.

Runtime Analysis

$\#$ maximal planar subsets of $K_f =$

Runtime Analysis

$\#$ maximal planar subsets of K_f $=$ $\#$ outerplanar graphs on k_f vertices

Runtime Analysis

$\#$ maximal planar subsets of K_f $=$ $\#$ outerplanar graphs on k_f vertices $< 4^{k_f}$.

Runtime Analysis

$\#$ maximal planar subsets of $K_f = \#$ outerplanar graphs on k_f vertices $< 4^{k_f}$.

$\#$ augmentations for the whole graph $\leq 4^k$

Runtime Analysis

$\#$ maximal planar subsets of $K_f = \#$ outerplanar graphs on k_f vertices $< 4^{k_f}$.

$\#$ augmentations for the whole graph $\leq 4^k$

Each augmentation consists of $\leq \sum_f 2k_f - 3 \leq 2k$ kitty corner edges.

Runtime Analysis

$\#$ maximal planar subsets of $K_f = \#$ outerplanar graphs on k_f vertices $< 4^{k_f}$.

$\#$ augmentations for the whole graph $\leq 4^k$

Each augmentation consists of $\leq \sum_f 2k_f - 3 \leq 2k$ kitty corner edges.

For each such edge, we have 3 possibilities to define D_x and 3 to define D_y.

Runtime Analysis

$\#$ maximal planar subsets of $K_f = \#$ outerplanar graphs on k_f vertices $< 4^{k_f}$.

$\#$ augmentations for the whole graph $\leq 4^k$

Each augmentation consists of $\leq \sum_f 2k_f - 3 \leq 2k$ kitty corner edges.

For each such edge, we have 3 possibilities to define D_x and 3 to define D_y.

$\Rightarrow |\mathcal{D}_x \times \mathcal{D}_y| \leq 3^{2k} \cdot 3^{2k} = 3^{4k} = 81^k$.

Runtime Analysis

$\#$ maximal planar subsets of $K_f = \#$ outerplanar graphs on k_f vertices $< 4^{k_f}$.

$\#$ augmentations for the whole graph $\leq 4^k$

Each augmentation consists of $\leq \sum_f 2k_f - 3 \leq 2k$ kitty corner edges.

For each such edge, we have 3 possibilities to define D_x and 3 to define D_y.

$\Rightarrow |\mathcal{D}_x \times \mathcal{D}_y| \leq 3^{2k} \cdot 3^{2k} = 3^{4k} = 81^k$.

Total running time $= \#$ augmentations $\cdot |\mathcal{D}_x \times \mathcal{D}_y| \cdot$ self-intersection check

Runtime Analysis

$\#$ maximal planar subsets of $K_f = \#$ outerplanar graphs on k_f vertices $< 4^{k_f}$.

$\#$ augmentations for the whole graph $\leq 4^k$

Each augmentation consists of $\leq \sum_f 2k_f - 3 \leq 2k$ kitty corner edges.

For each such edge, we have 3 possibilities to define D_x and 3 to define D_y.

$\Rightarrow |\mathcal{D}_x \times \mathcal{D}_y| \leq 3^{2k} \cdot 3^{2k} = 3^{4k} = 81^k$.

Total running time $=$ $\#$ augmentations $\cdot |\mathcal{D}_x \times \mathcal{D}_y| \cdot$ self-intersection check

Runtime Analysis

$\#$ maximal planar subsets of K_f $=$ $\#$ outerplanar graphs on k_f vertices $< 4^{k_f}$.

$\#$ augmentations for the whole graph $\leq 4^k$

Each augmentation consists of $\leq \sum_f 2k_f - 3 \leq 2k$ kitty corner edges.

For each such edge, we have 3 possibilities to define D_x and 3 to define D_y.

$\Rightarrow |\mathcal{D}_x \times \mathcal{D}_y| \leq 3^{2k} \cdot 3^{2k} = 3^{4k} = 81^k$.

Total running time $=$ $\#$ augmentations $\cdot |\mathcal{D}_x \times \mathcal{D}_y| \cdot$ self-intersection check

$\leq 324^k \cdot O(n \log n)$

Runtime Analysis

$\#$ maximal planar subsets of K_f = $\#$ outerplanar graphs on k_f vertices $< 4^{k_f}$.

$\#$ augmentations for the whole graph $\leq 4^k$

Each augmentation consists of $\leq \sum_f 2k_f - 3 \leq 2k$ kitty corner edges.

For each such edge, we have 3 possibilities to define D_x and 3 to define D_y.

$\Rightarrow |\mathcal{D}_x \times \mathcal{D}_y| \leq 3^{2k} \cdot 3^{2k} = 3^{4k} = 81^k$.

Total running time = $\#$ augmentations $\cdot |\mathcal{D}_x \times \mathcal{D}_y| \cdot$ self-intersection check

$\leq 324^k \cdot O(n \log n)$

by a sweep-line algorithm, but we think that $O(n)$ suffices.

Runtime Analysis

$\#$ maximal planar subsets of K_f $=$ $\#$ outerplanar graphs on k_f vertices $< 4^{k_f}$.

$\#$ augmentations for the whole graph $\leq 4^k$

Each augmentation consists of $\leq \sum_f 2k_f - 3 \leq 2k$ kitty corner edges.

For each such edge, we have 3 possibilities to define D_x and 3 to define D_y.

$\Rightarrow |\mathcal{D}_x \times \mathcal{D}_y| \leq 3^{2k} \cdot 3^{2k} = 3^{4k} = 81^k$.

Total running time $=$ $\#$ augmentations $\cdot |\mathcal{D}_x \times \mathcal{D}_y| \cdot$ self-intersection check

$\leq 324^k \cdot O(n \log n)$

by a sweep-line algorithm, but we think that $O(n)$ suffices.

Theorem. OC is FPT w.r.t. the number of kitty corners.

Table of Contents

- **Number of kitty corners:** (the number of corners involved in some pair of kitty corners)
  We show: *OC admits an FPT algorithm parametrized by the number of kitty corners.*

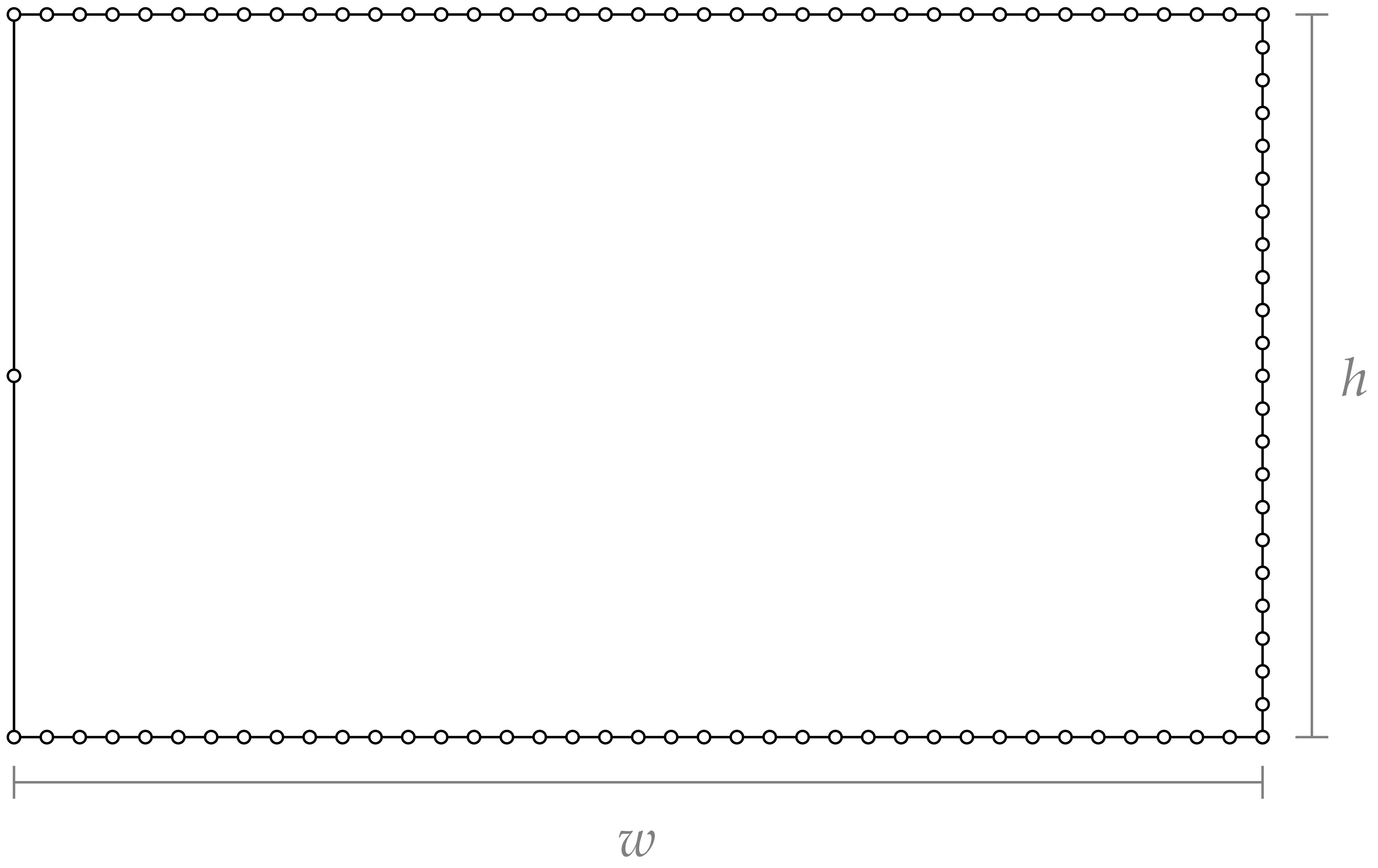
- **Number of faces:** OC is para-NP-hard when parametrized by the number of faces.
  For *one* face (cycle), we show the *existence of a polynomial kernel for OC*.
  when parametrized by the number of kitty corners.

- **Maximum face-degree:** The reductions of Patrignani & Evans et al. use linear-size faces.
  We show: *OC remains NP-hard when parametrized by maximum face degree.*

- **Height:** (minimum number of distinct y-coordinates required to draw the representation)
  A $(w \times h)$-grid has pathwidth at most h.
  $\Rightarrow$ Graphs of bounded height have bounded pathwidth. But converse generally not true.
  We show: *OC admits an XP algorithm parametrized by height.*

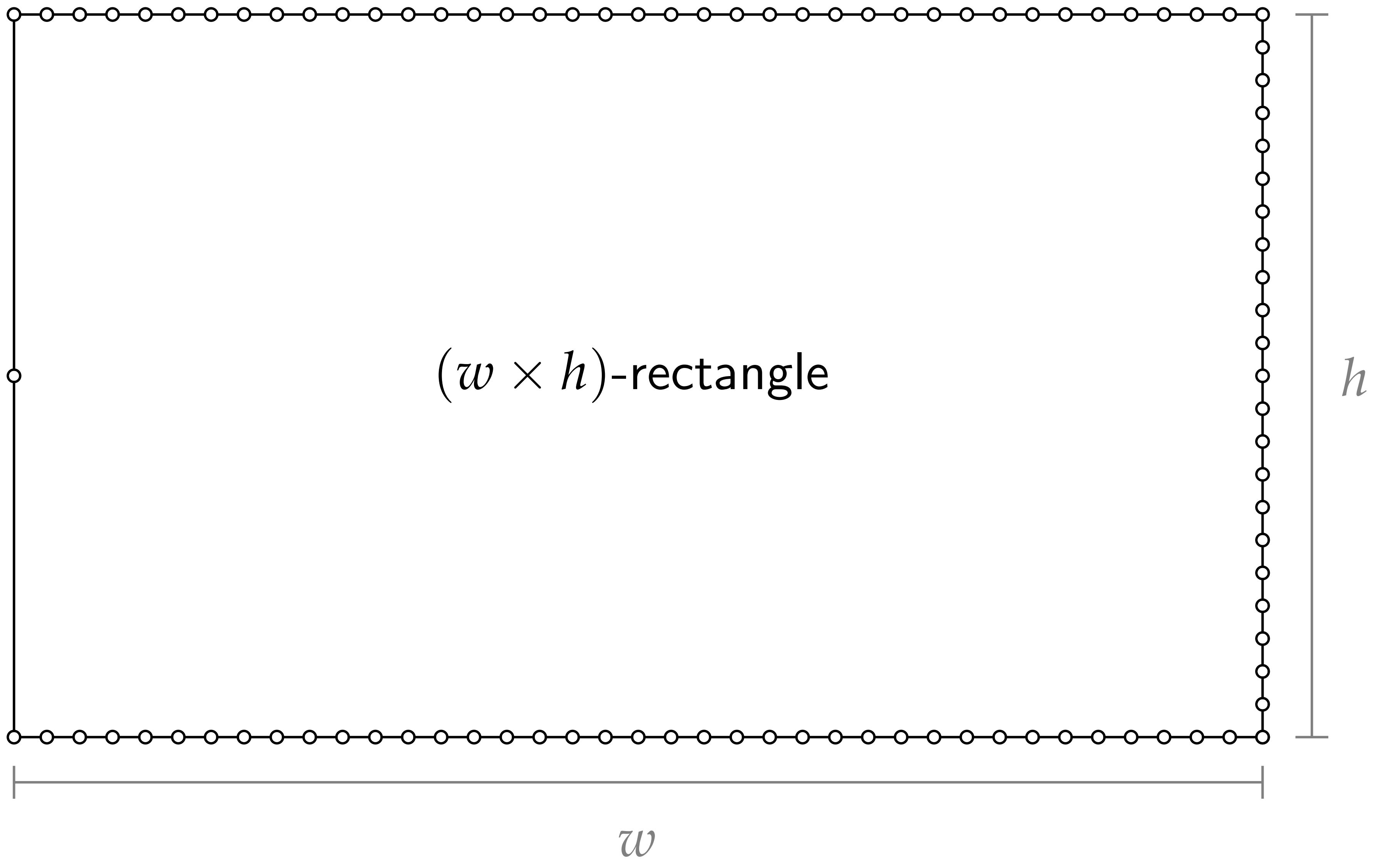
Boundary, Belt, and “Piston” Gadget

Theorem. OC is NP-hard. [Patrignani, CGTA 2001]



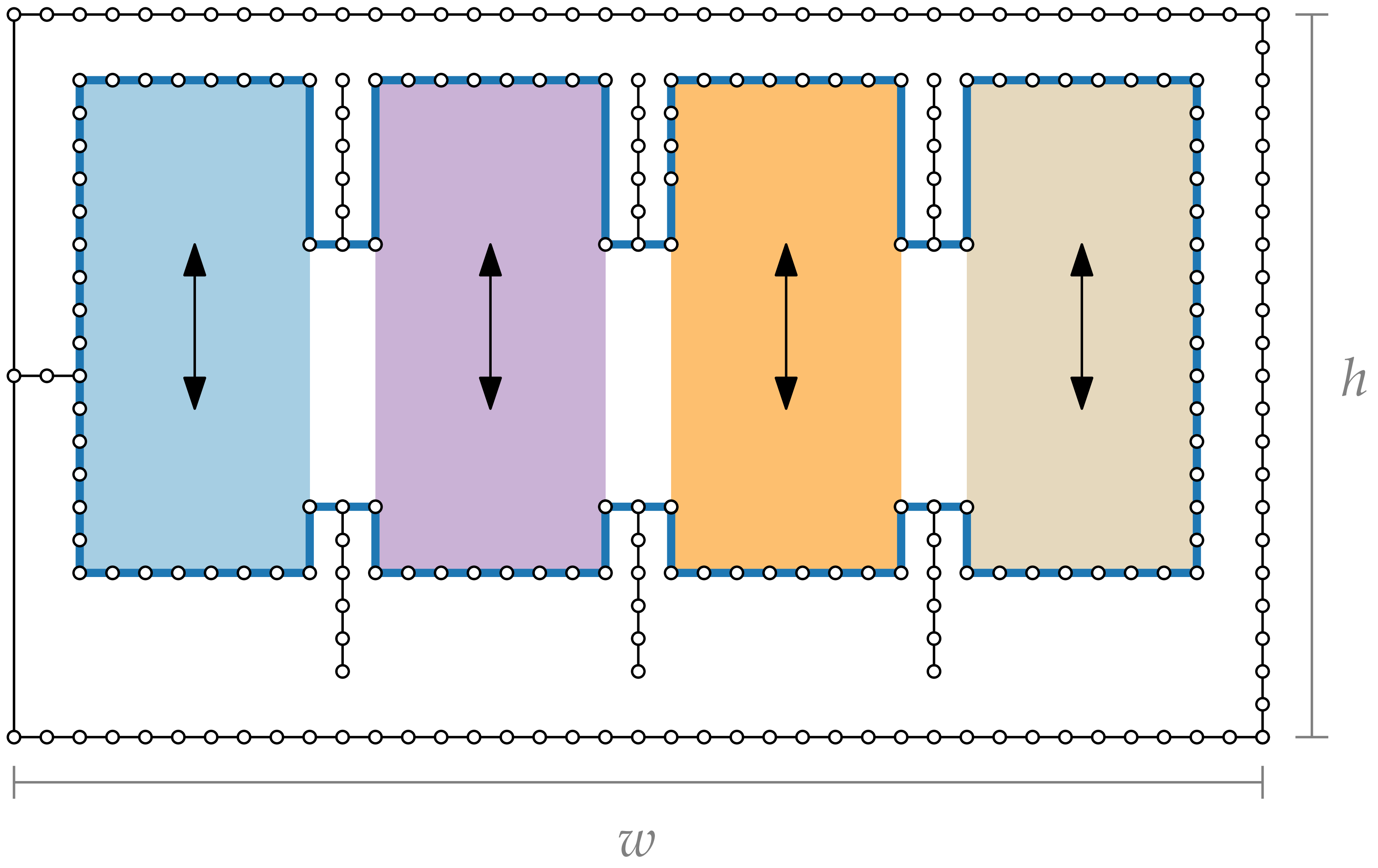
Boundary, Belt, and “Piston” Gadget

Theorem. OC is NP-hard. [Patrignani, CGTA 2001]



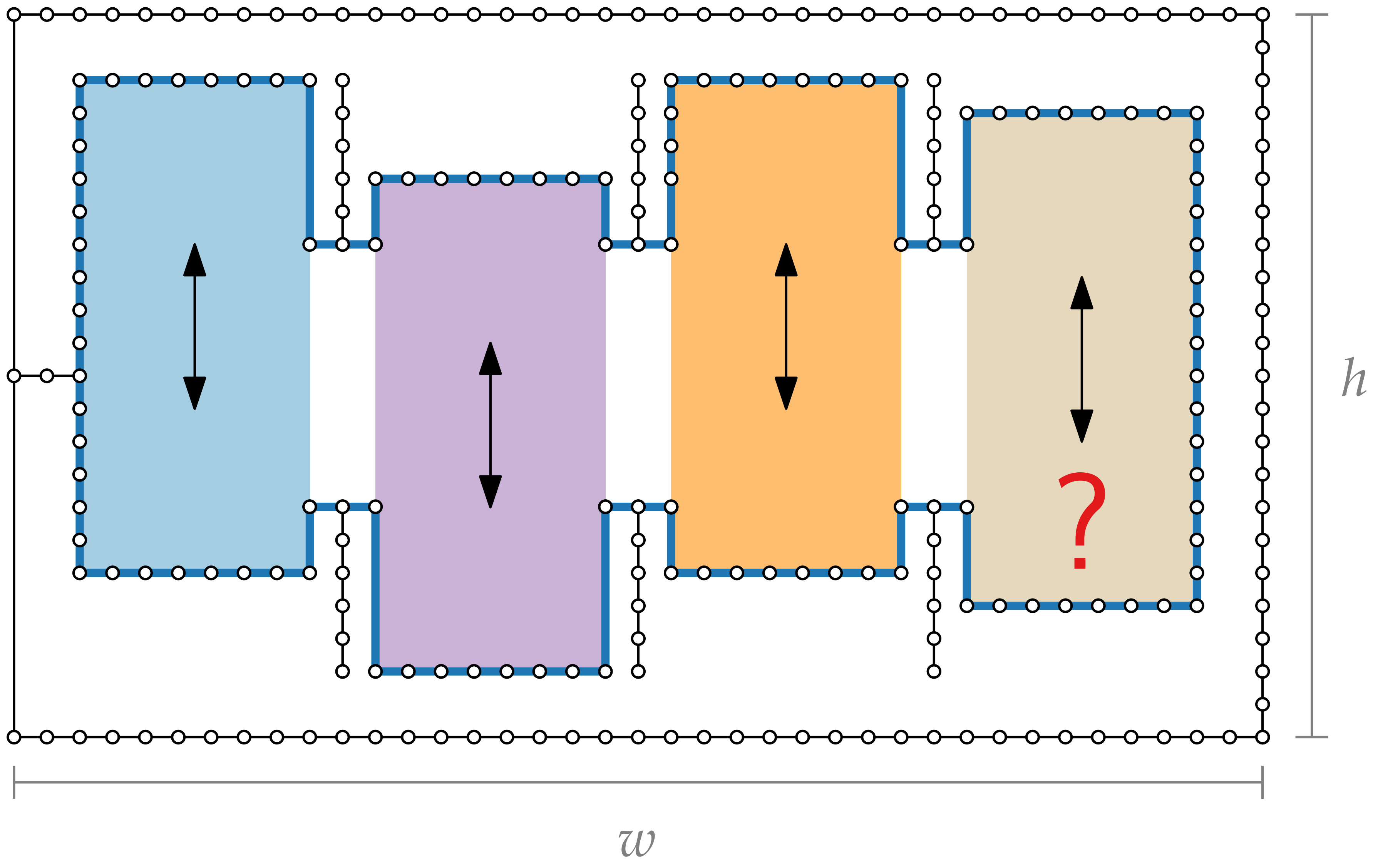
Boundary, Belt, and “Piston” Gadget

Theorem. OC is NP-hard. [Patrignani, CGTA 2001]



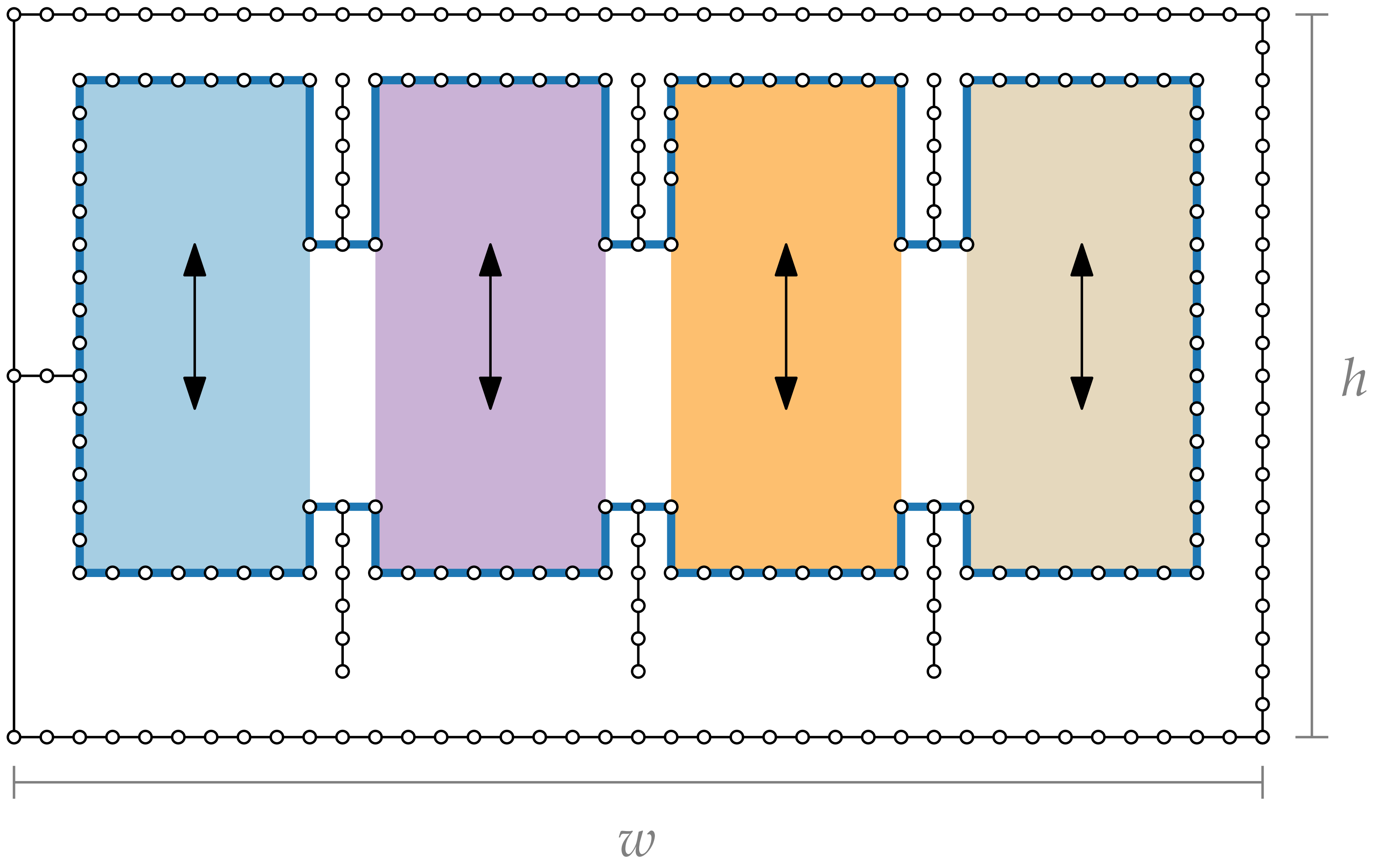
Boundary, Belt, and “Piston” Gadget

Theorem. OC is NP-hard. [Patrignani, CGTA 2001]



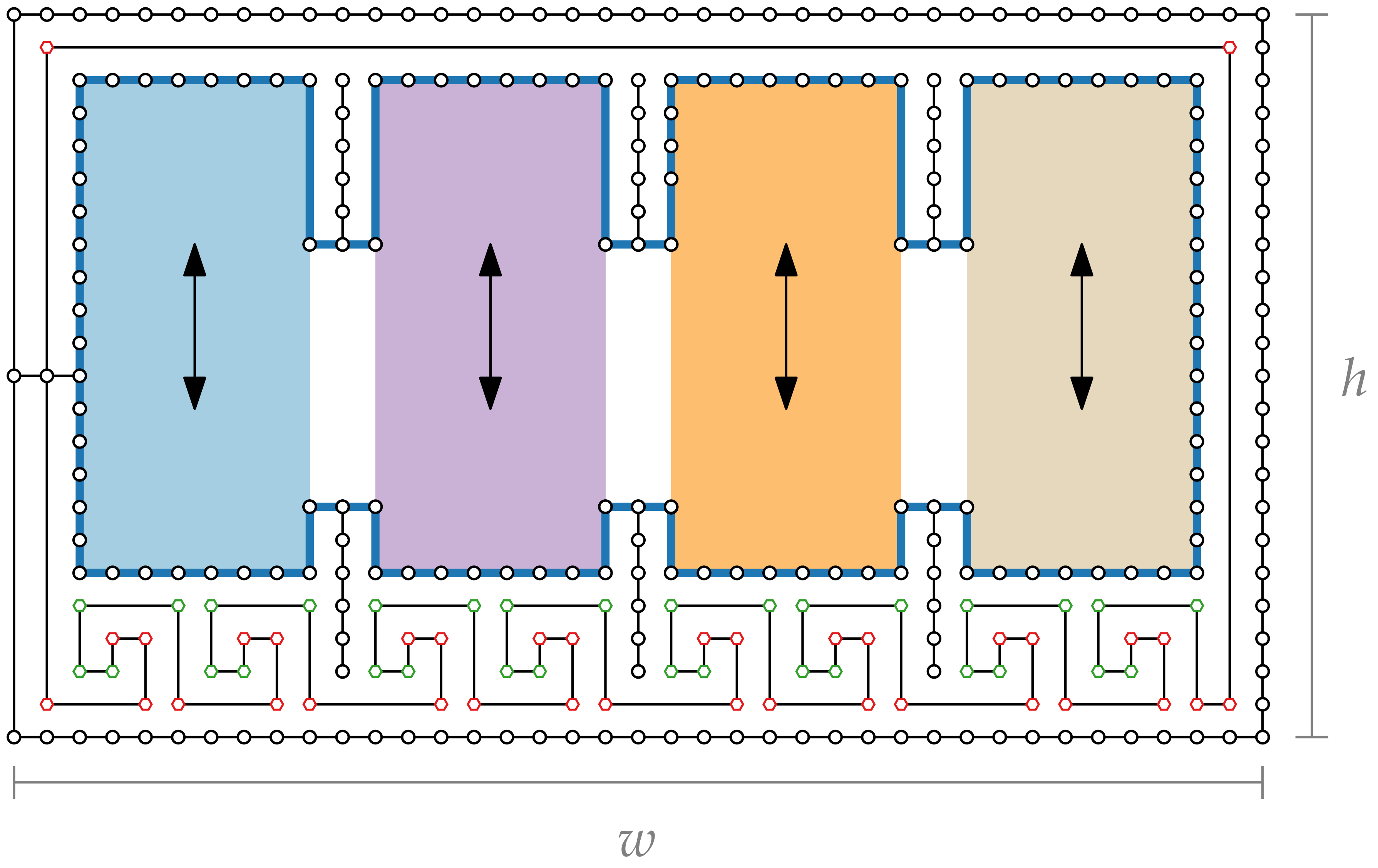
Boundary, Belt, and “Piston” Gadget

Theorem. OC is NP-hard. [Patrignani, CGTA 2001]



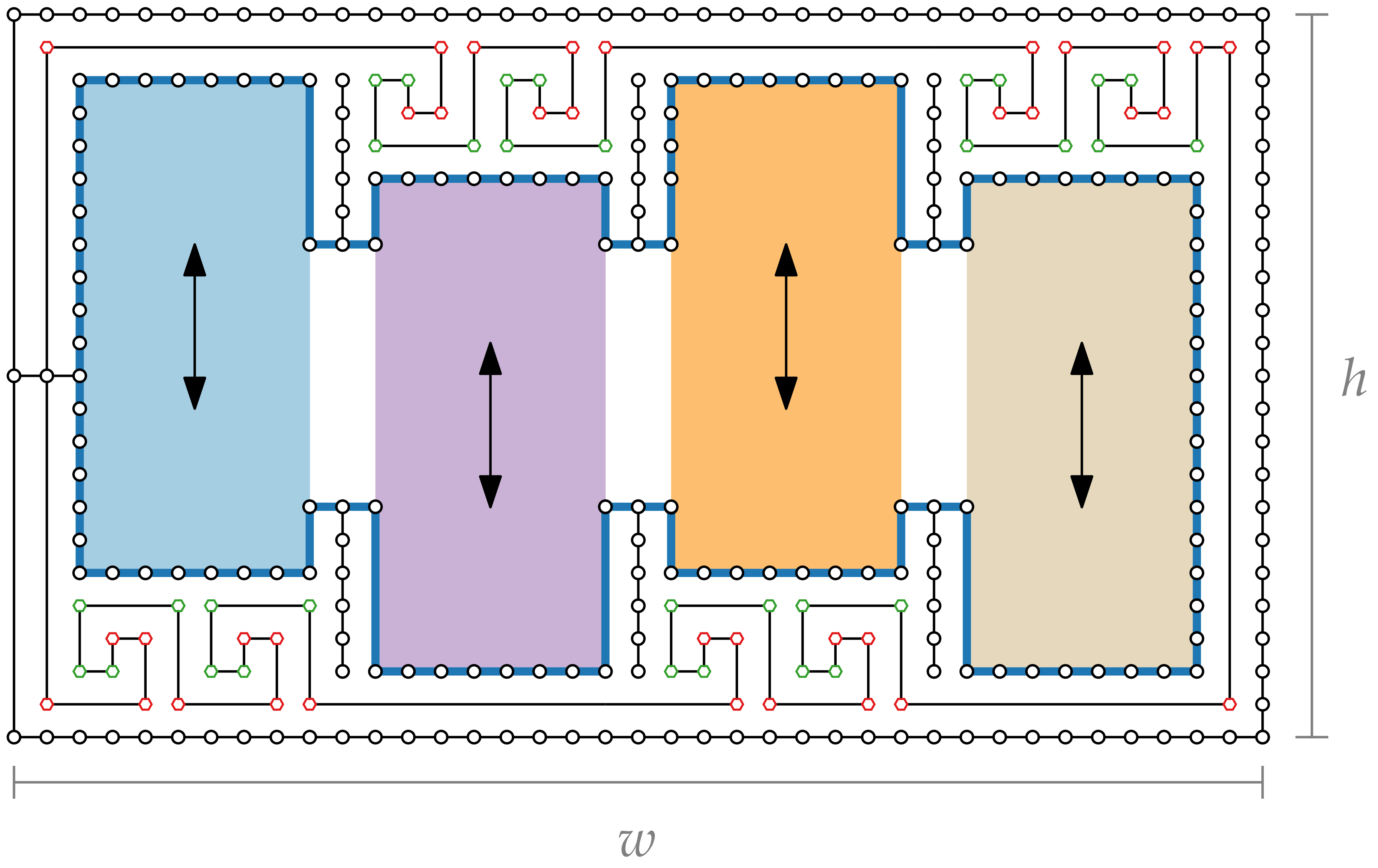
Boundary, Belt, and “Piston” Gadget

Theorem. OC is NP-hard. [Patrignani, CGTA 2001]



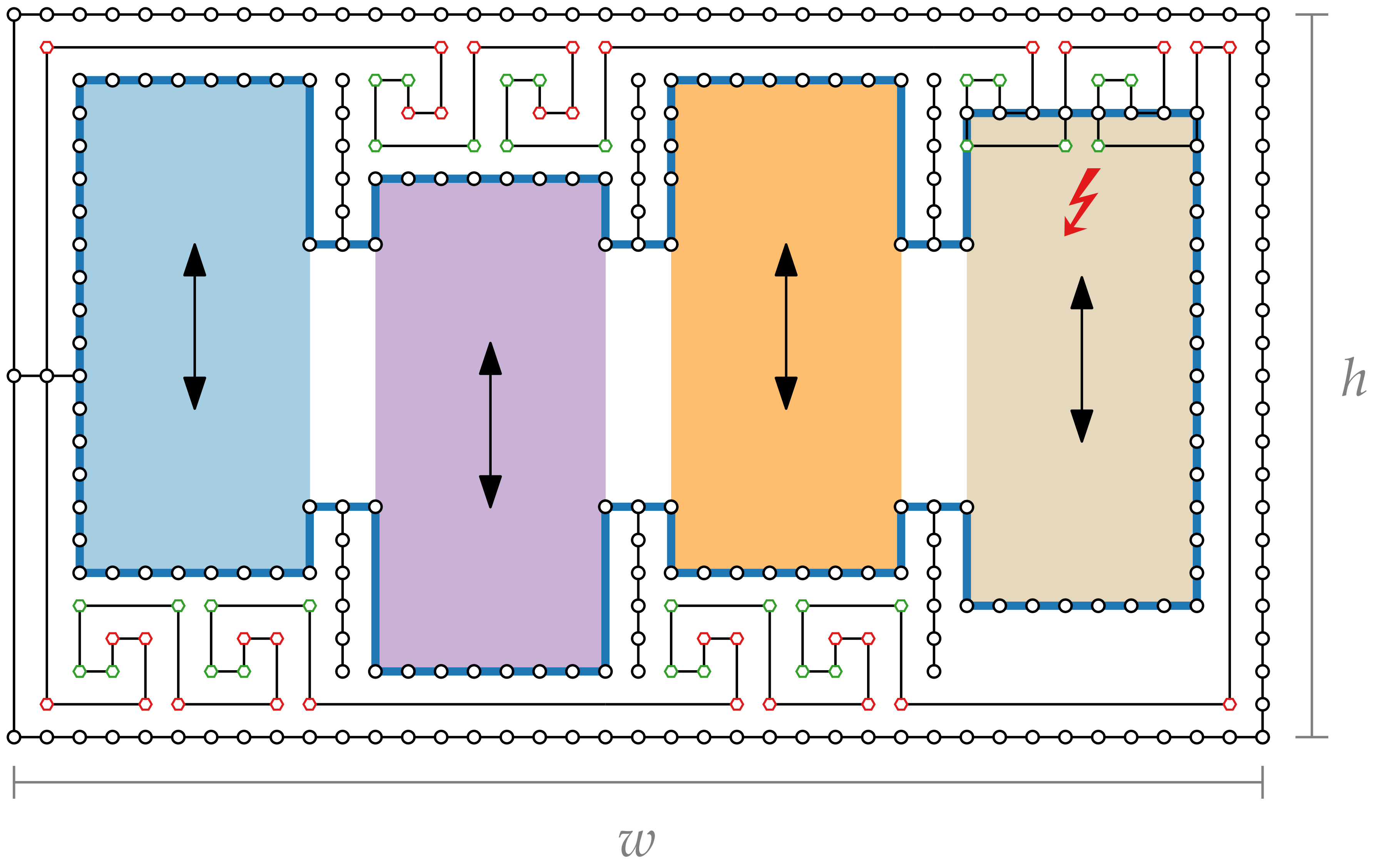
Boundary, Belt, and “Piston” Gadget

Theorem. OC is NP-hard. [Patrignani, CGTA 2001]



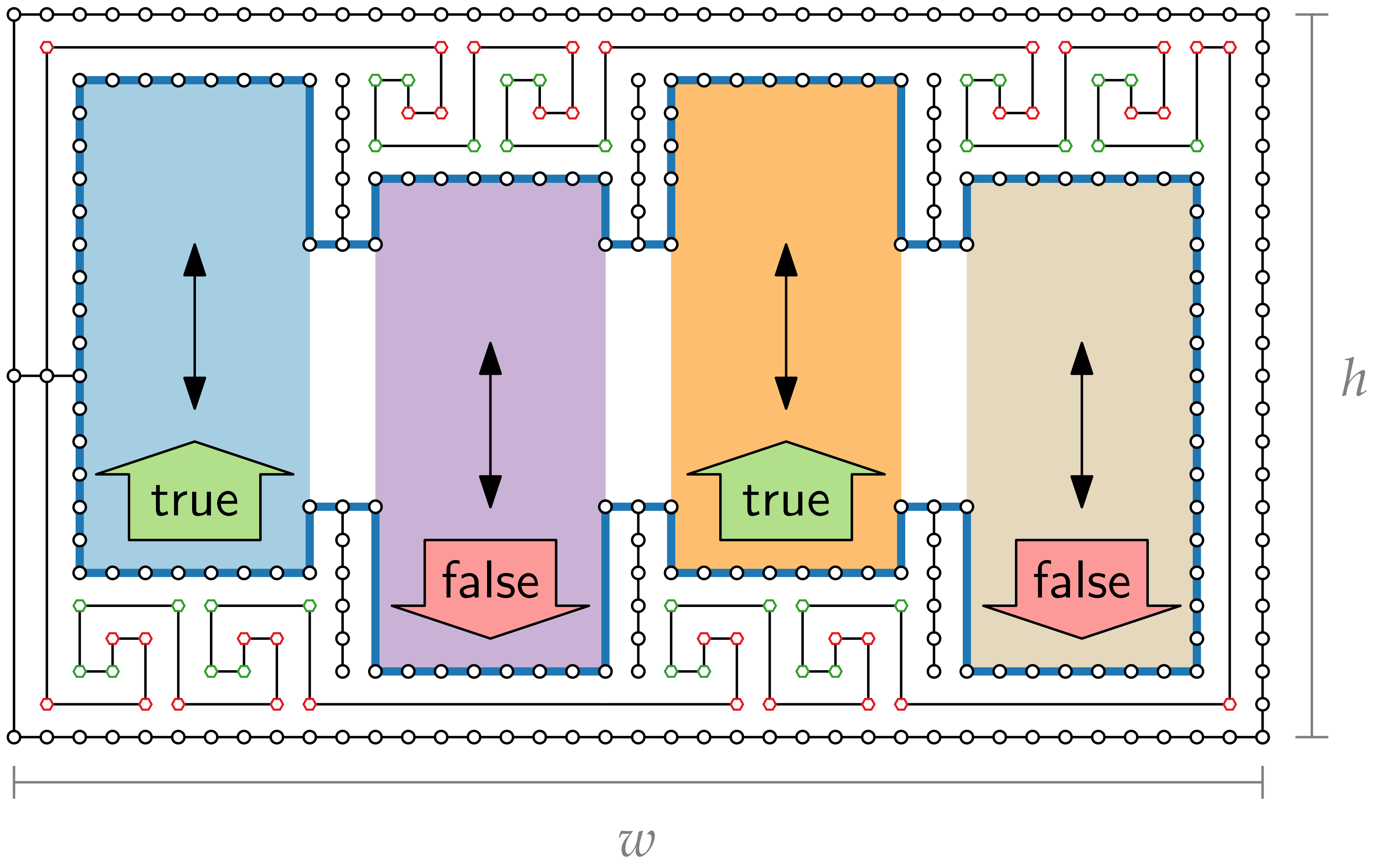
Boundary, Belt, and “Piston” Gadget

Theorem. OC is NP-hard. [Patrignani, CGTA 2001]

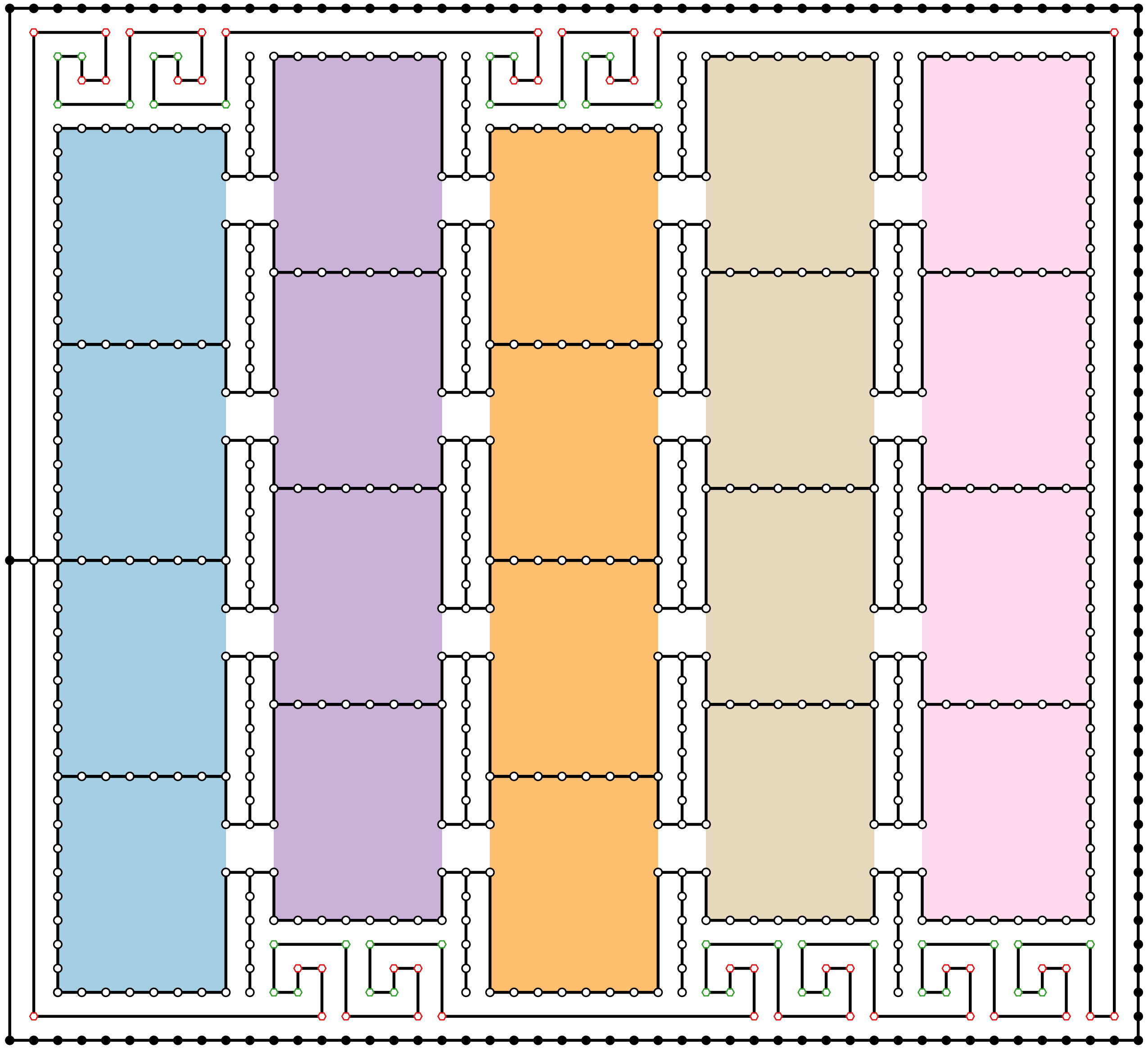


Boundary, Belt, and “Piston” Gadget

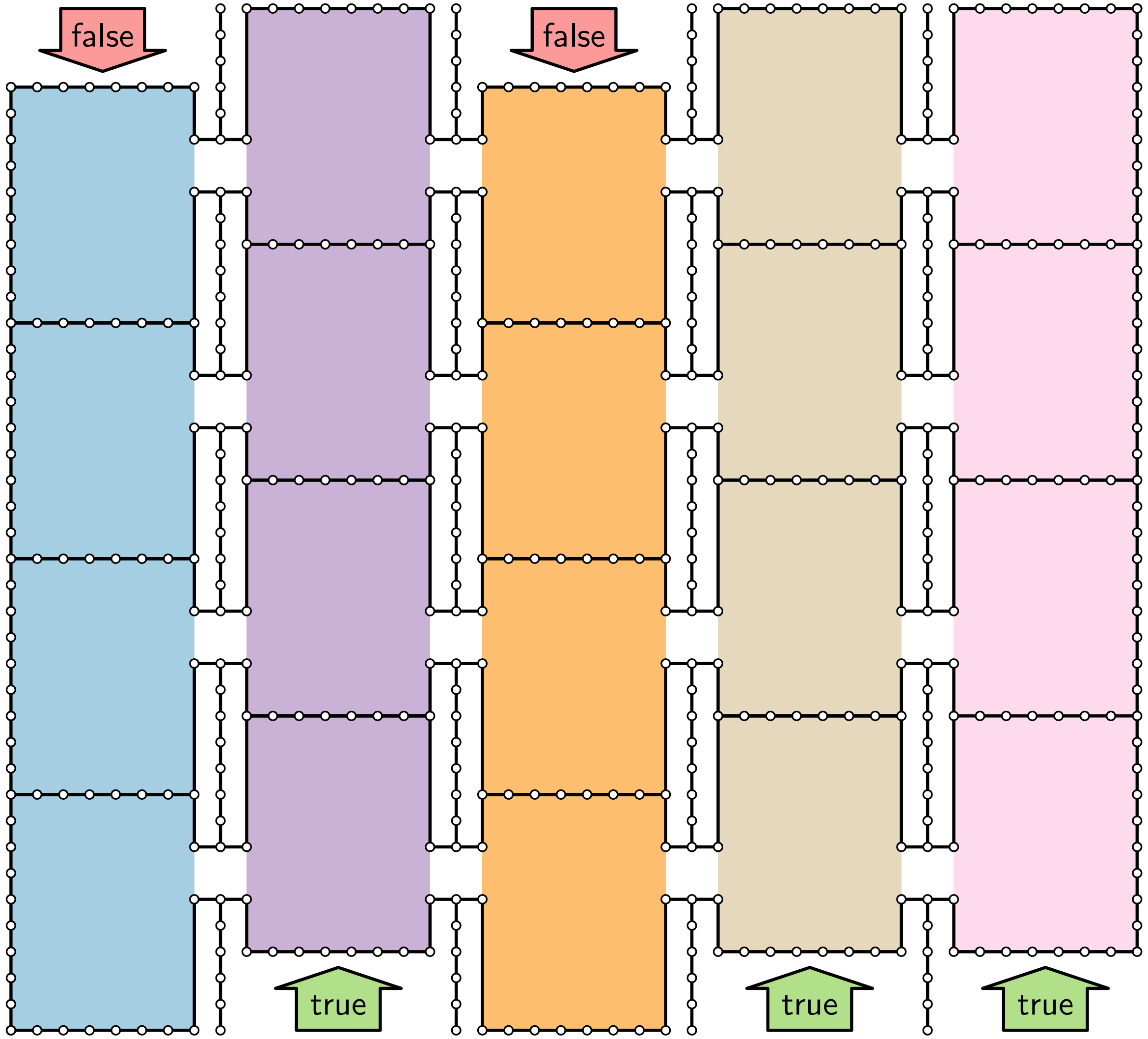
Theorem. OC is NP-hard. [Patrignani, CGTA 2001]



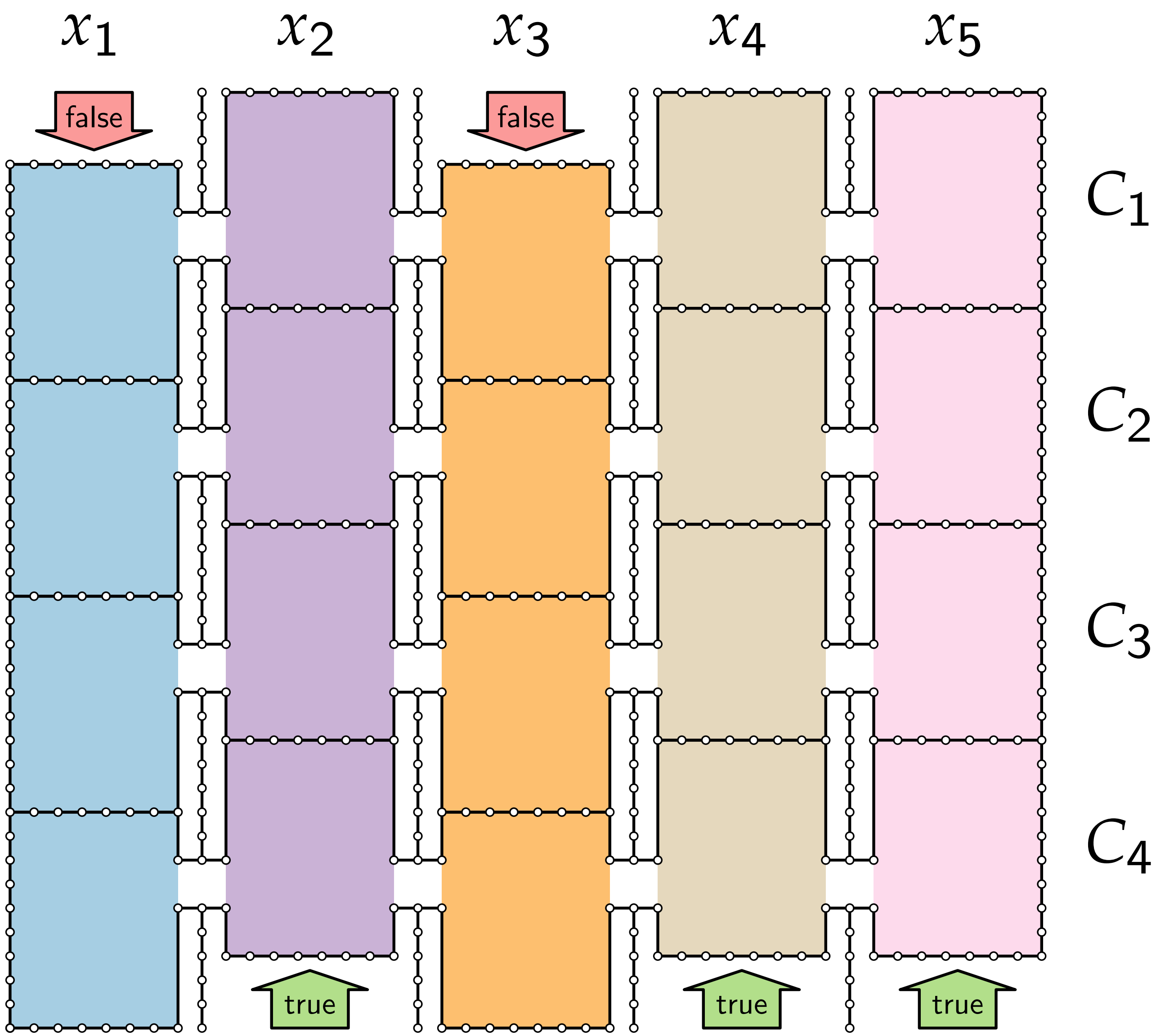
Clause Gadgets



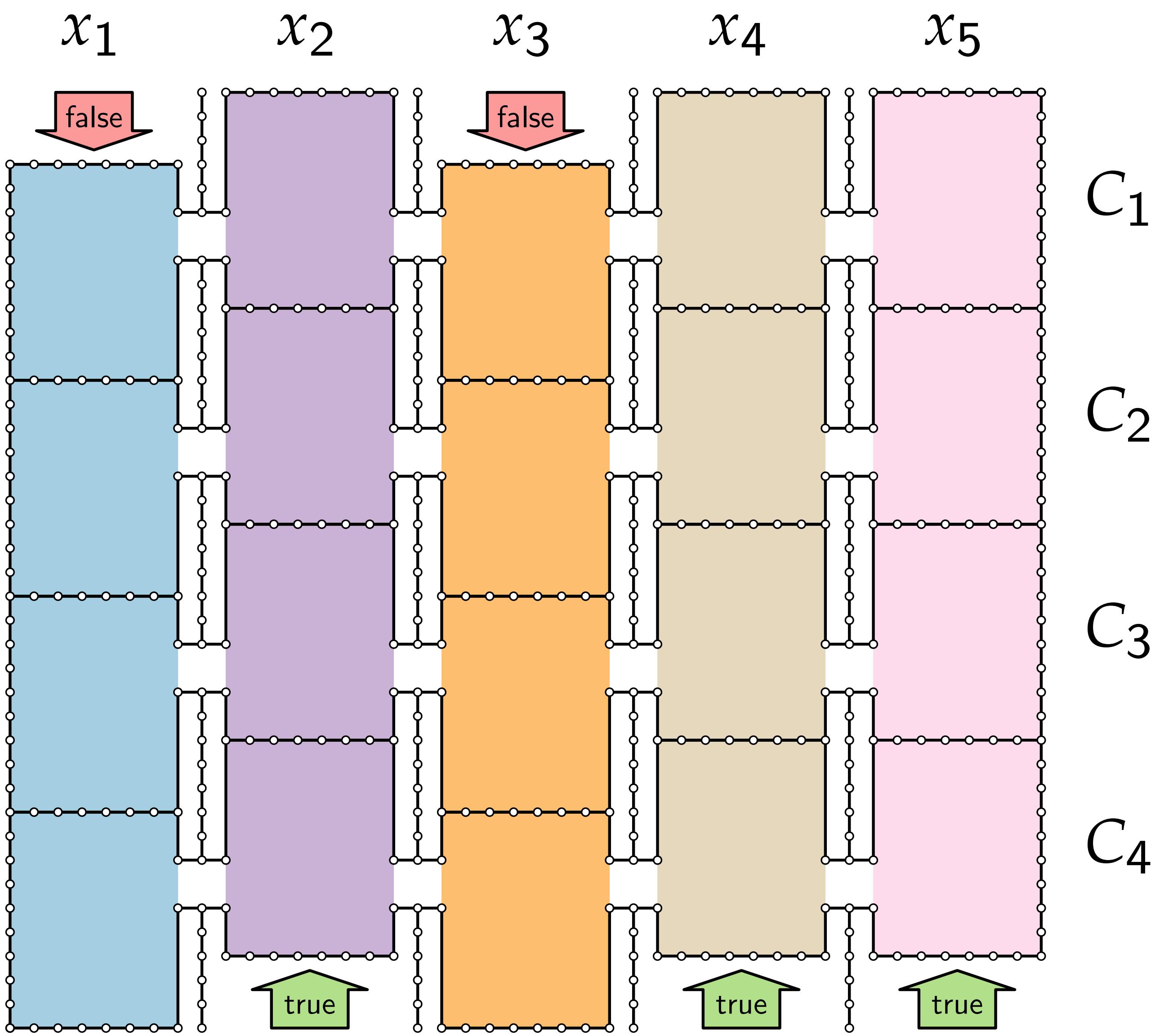
Clause Gadgets



Clause Gadgets



Clause Gadgets



Example:

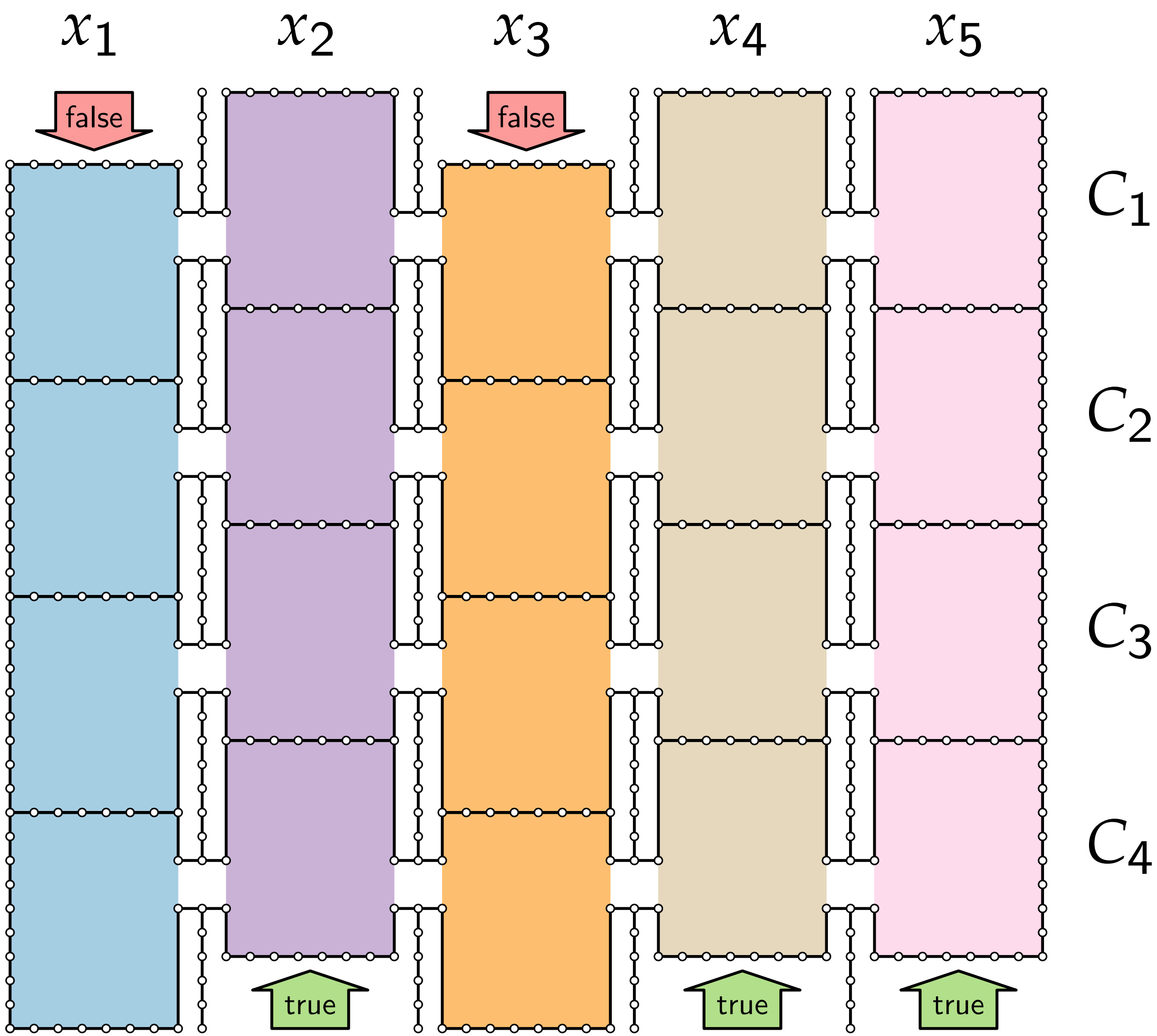
$C_1 = x_2 \vee \overline{x_4}$

$C_2 = x_1 \vee x_2 \vee \overline{x_3}$

$C_3 = x_5$

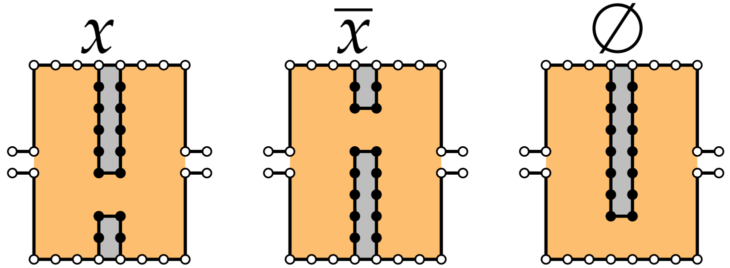
$C_4 = x_4 \vee \overline{x_5}$

Clause Gadgets

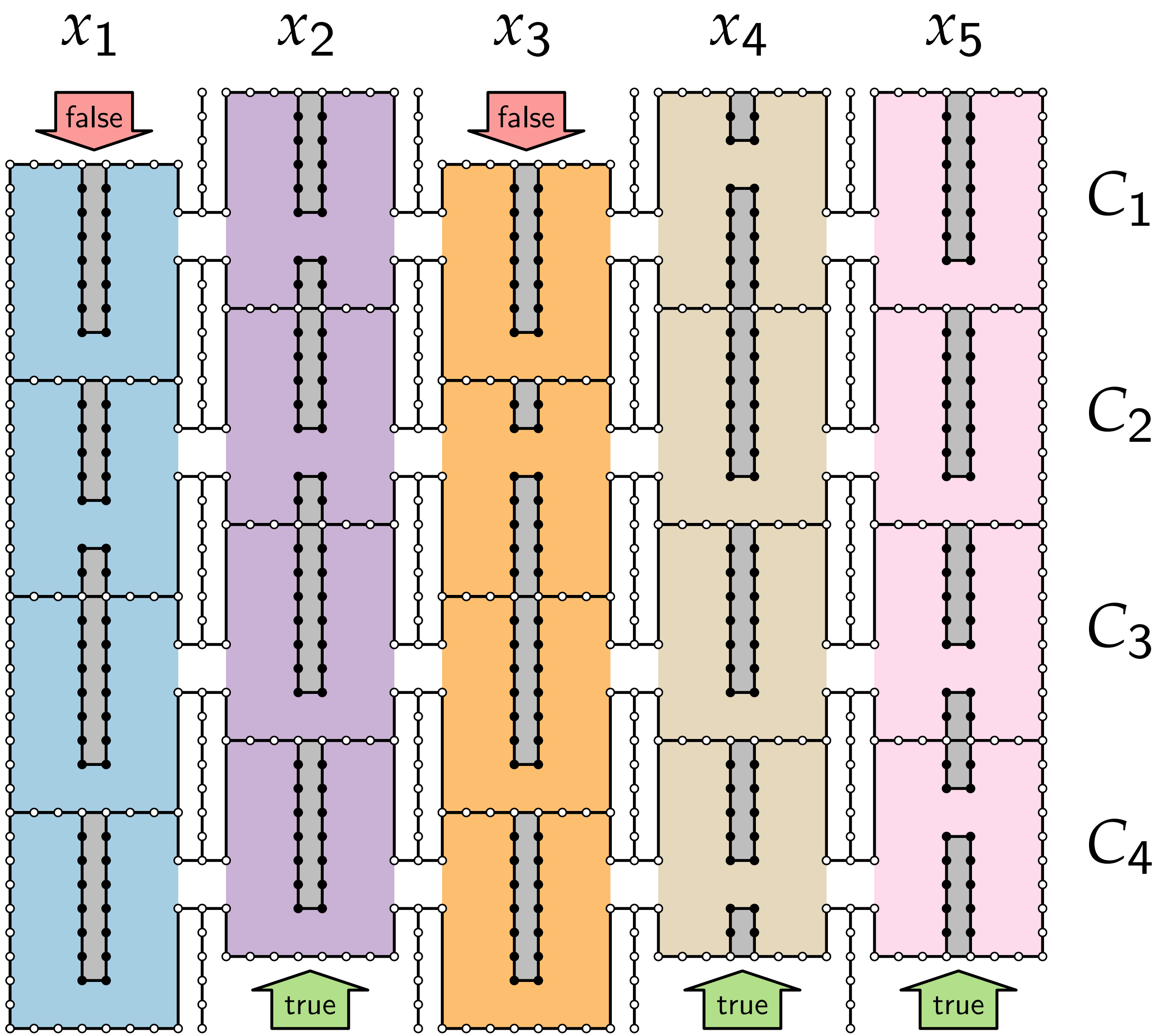


Example:

$C_1 = x_2 \vee \overline{x_4}$
$C_2 = x_1 \vee x_2 \vee \overline{x_3}$
$C_3 = x_5$
$C_4 = x_4 \vee \overline{x_5}$



Clause Gadgets



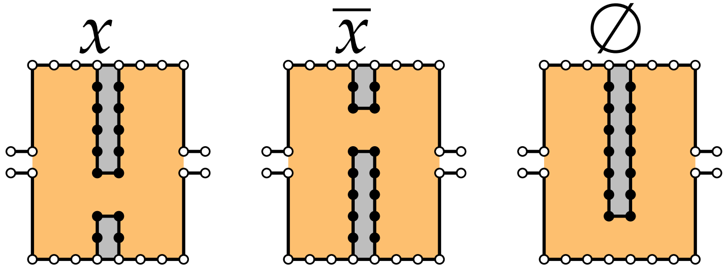
Example:

$C_1 = x_2 \vee \overline{x_4}$

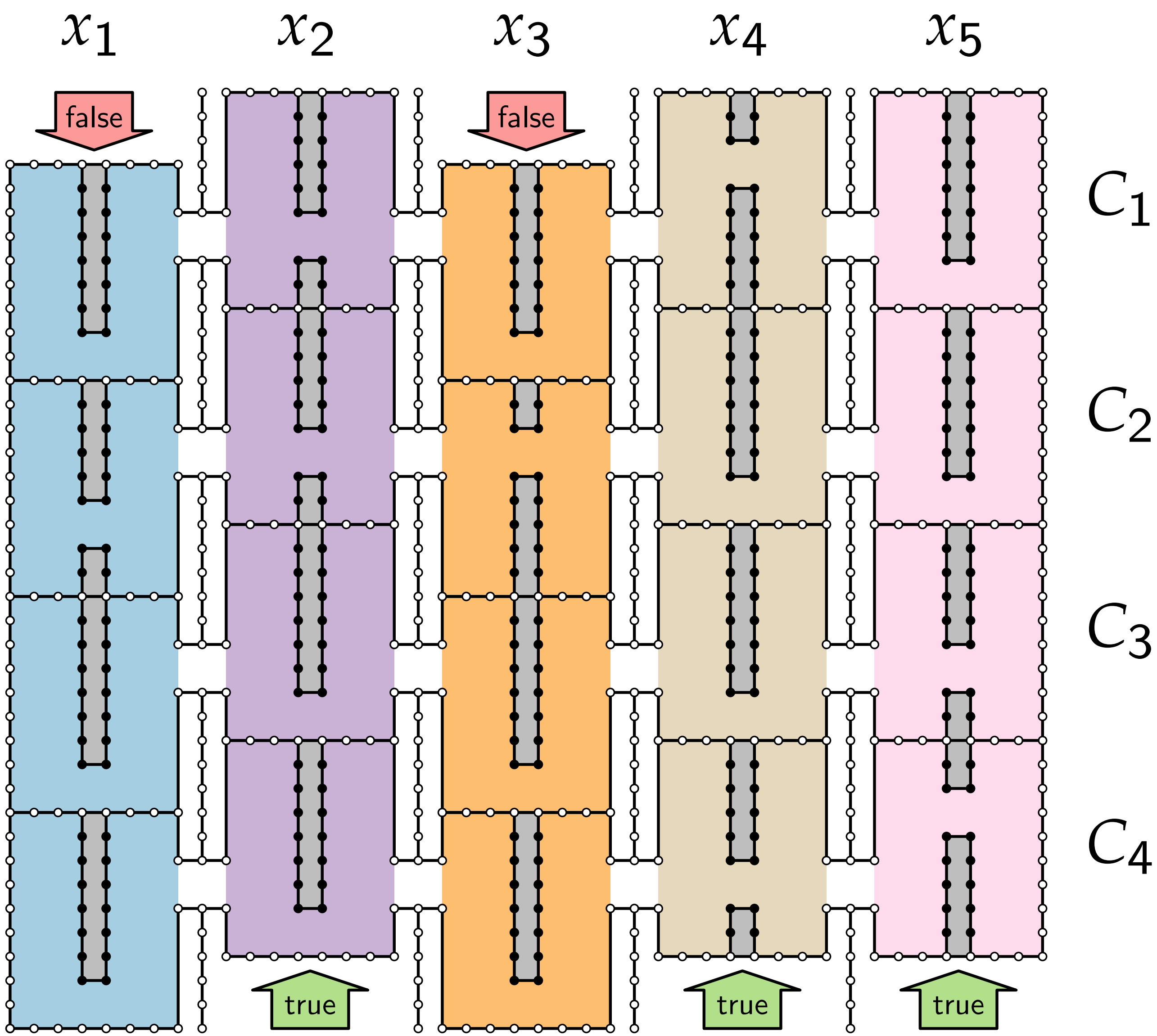
$C_2 = x_1 \vee x_2 \vee \overline{x_3}$

$C_3 = x_5$

$C_4 = x_4 \vee \overline{x_5}$

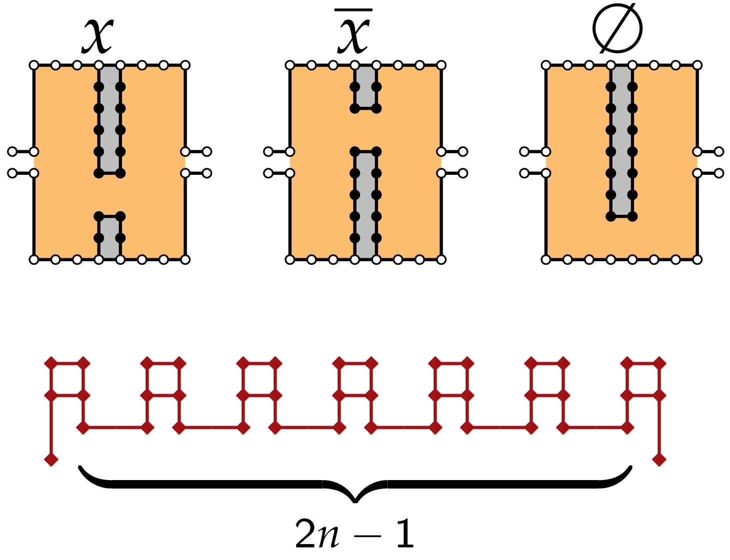


Clause Gadgets

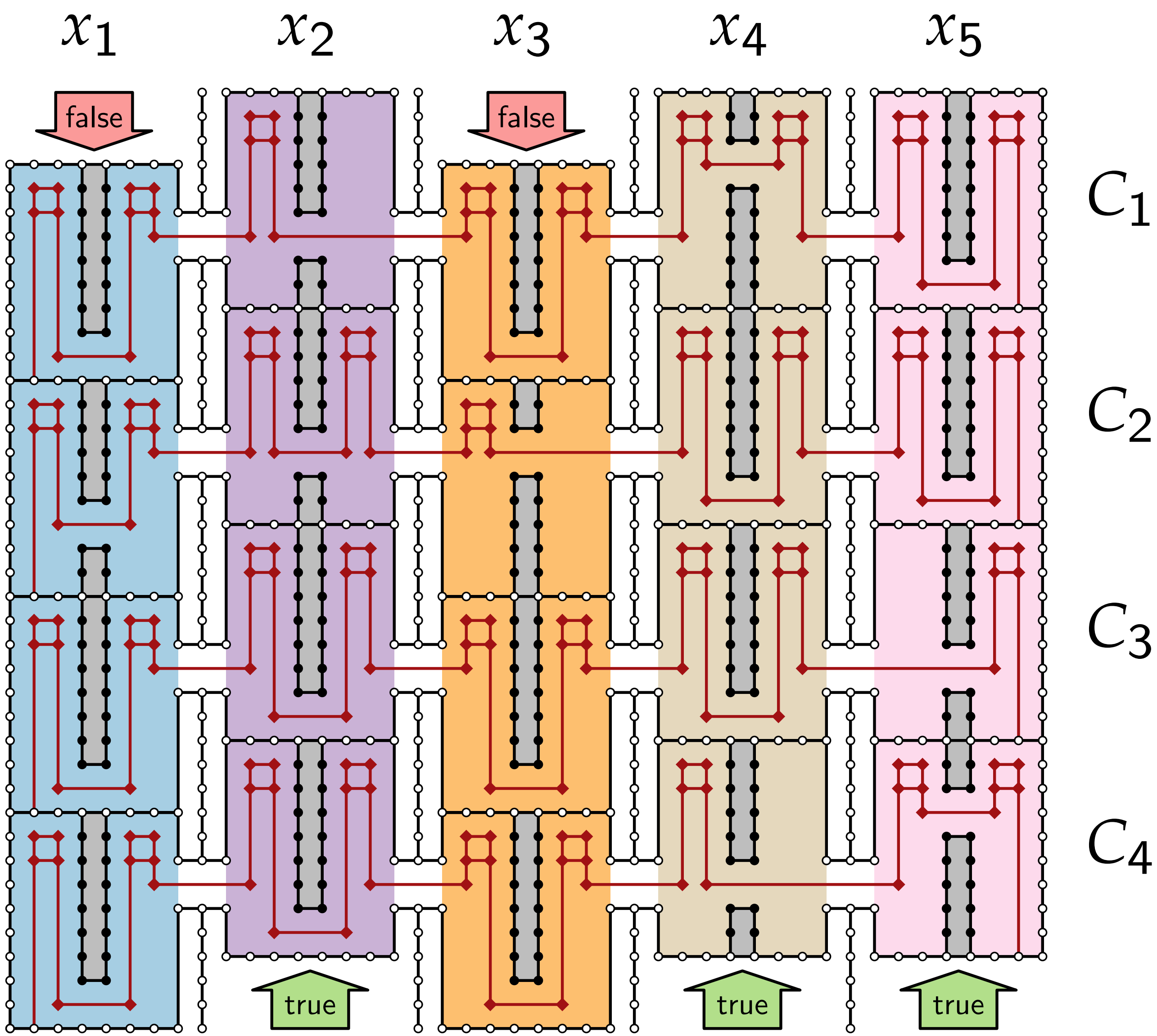


Example:

$C_1 = x_2 \vee \overline{x_4}$
$C_2 = x_1 \vee x_2 \vee \overline{x_3}$
$C_3 = x_5$
$C_4 = x_4 \vee \overline{x_5}$



Clause Gadgets



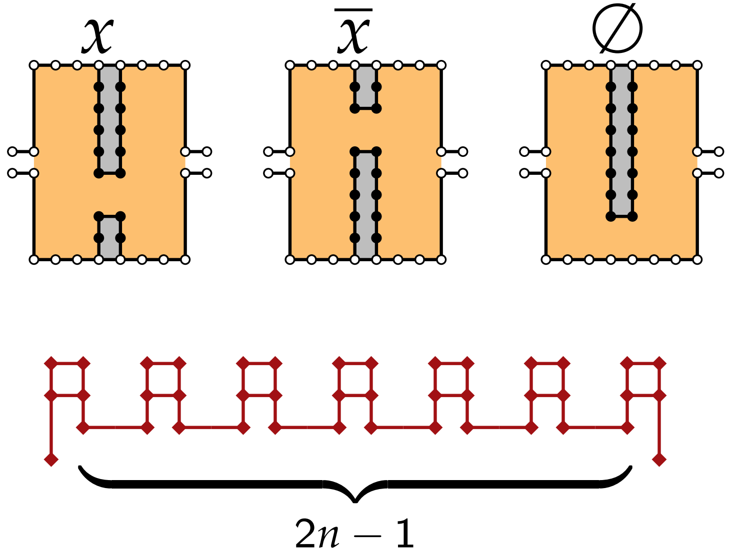
Example:

$C_1 = x_2 \vee \overline{x_4}$

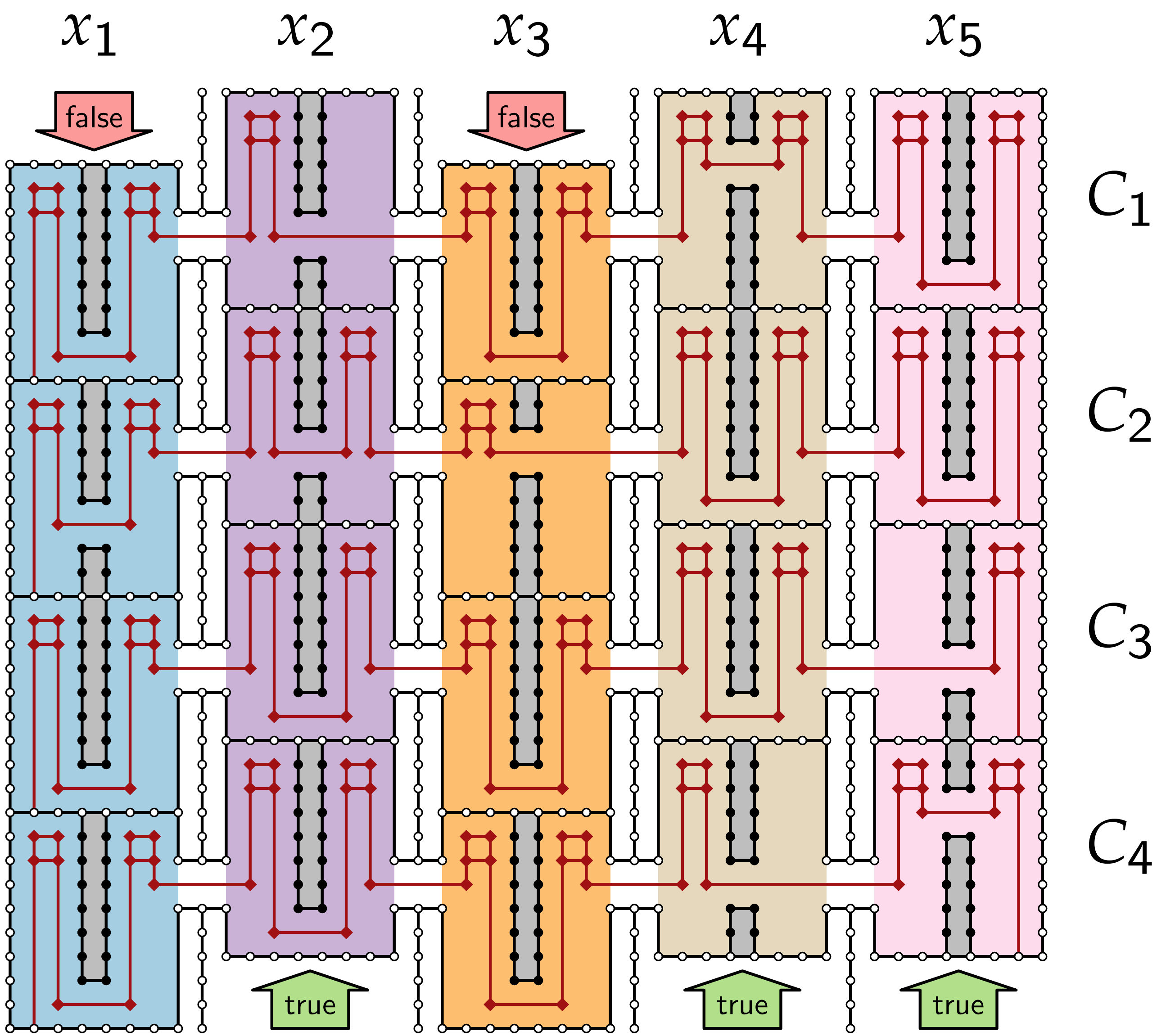
$C_2 = x_1 \vee x_2 \vee \overline{x_3}$

$C_3 = x_5$

$C_4 = x_4 \vee \overline{x_5}$



Clause Gadgets



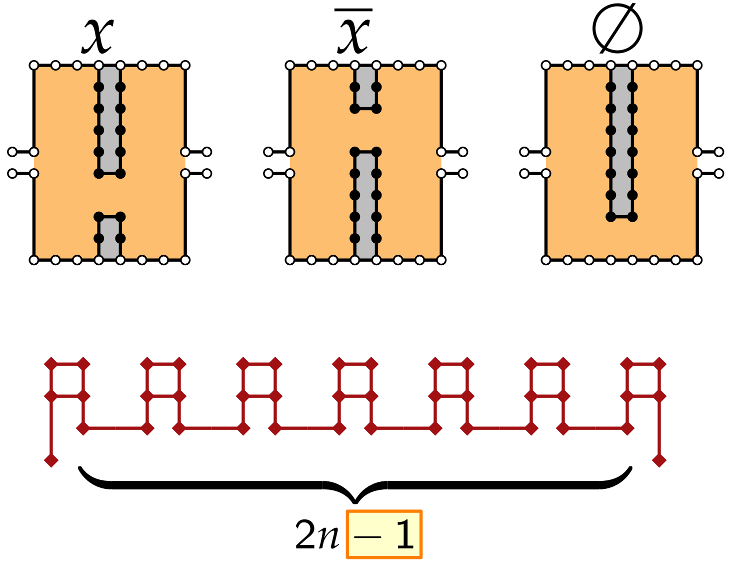
Example:

$C_1 = x_2 \vee \overline{x_4}$

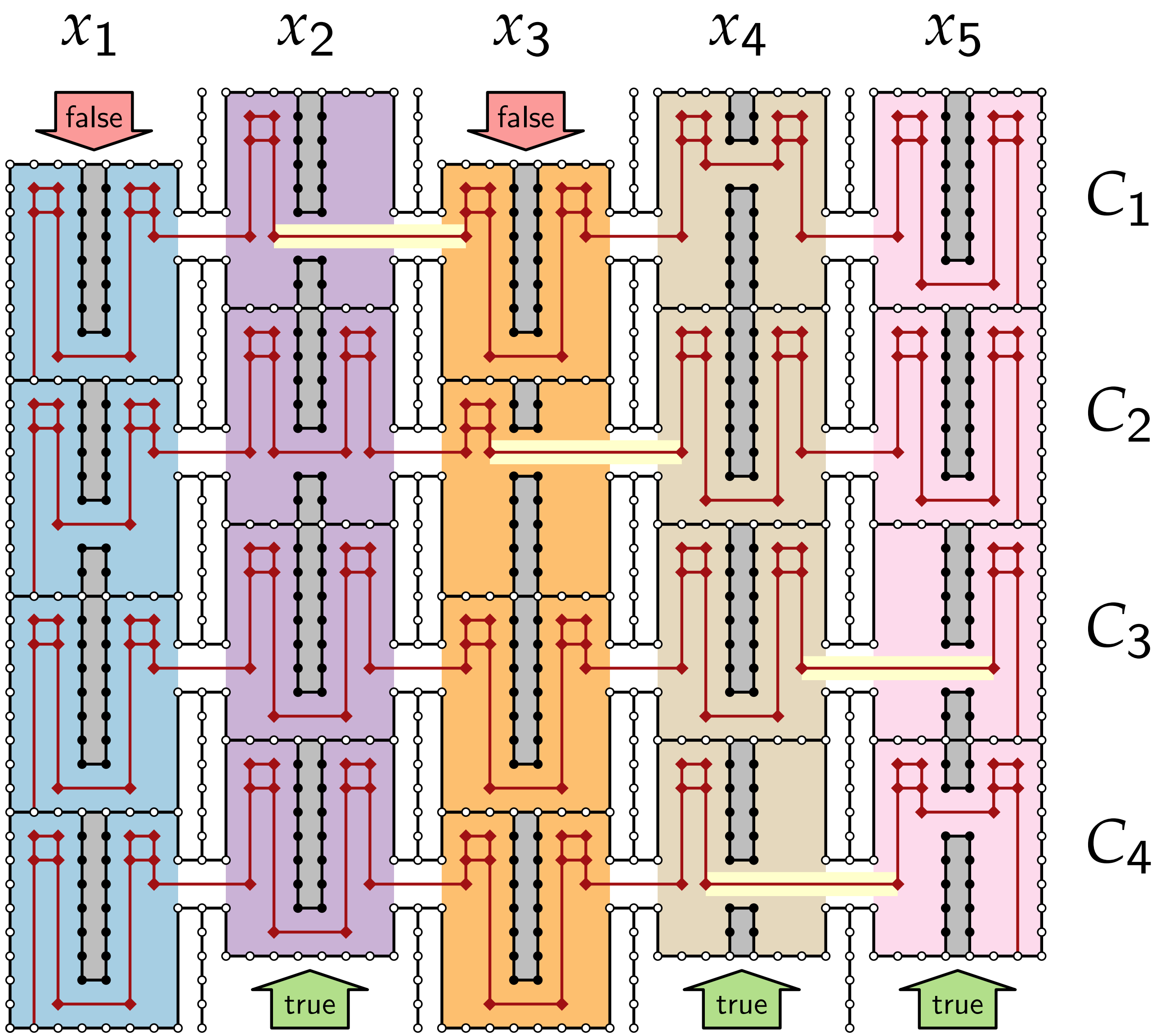
$C_2 = x_1 \vee x_2 \vee \overline{x_3}$

$C_3 = x_5$

$C_4 = x_4 \vee \overline{x_5}$



Clause Gadgets



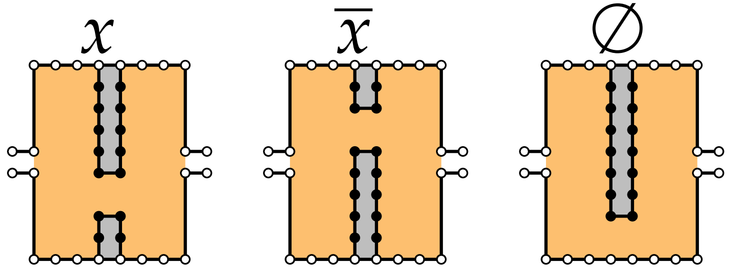
Example:

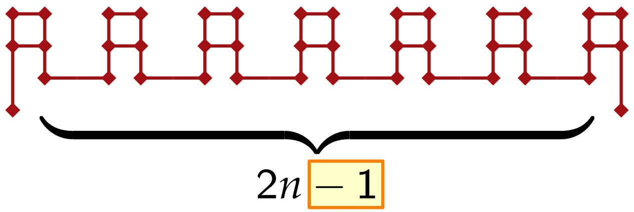
$C_1 = x_2 \vee \overline{x_4}$

$C_2 = x_1 \vee x_2 \vee \overline{x_3}$

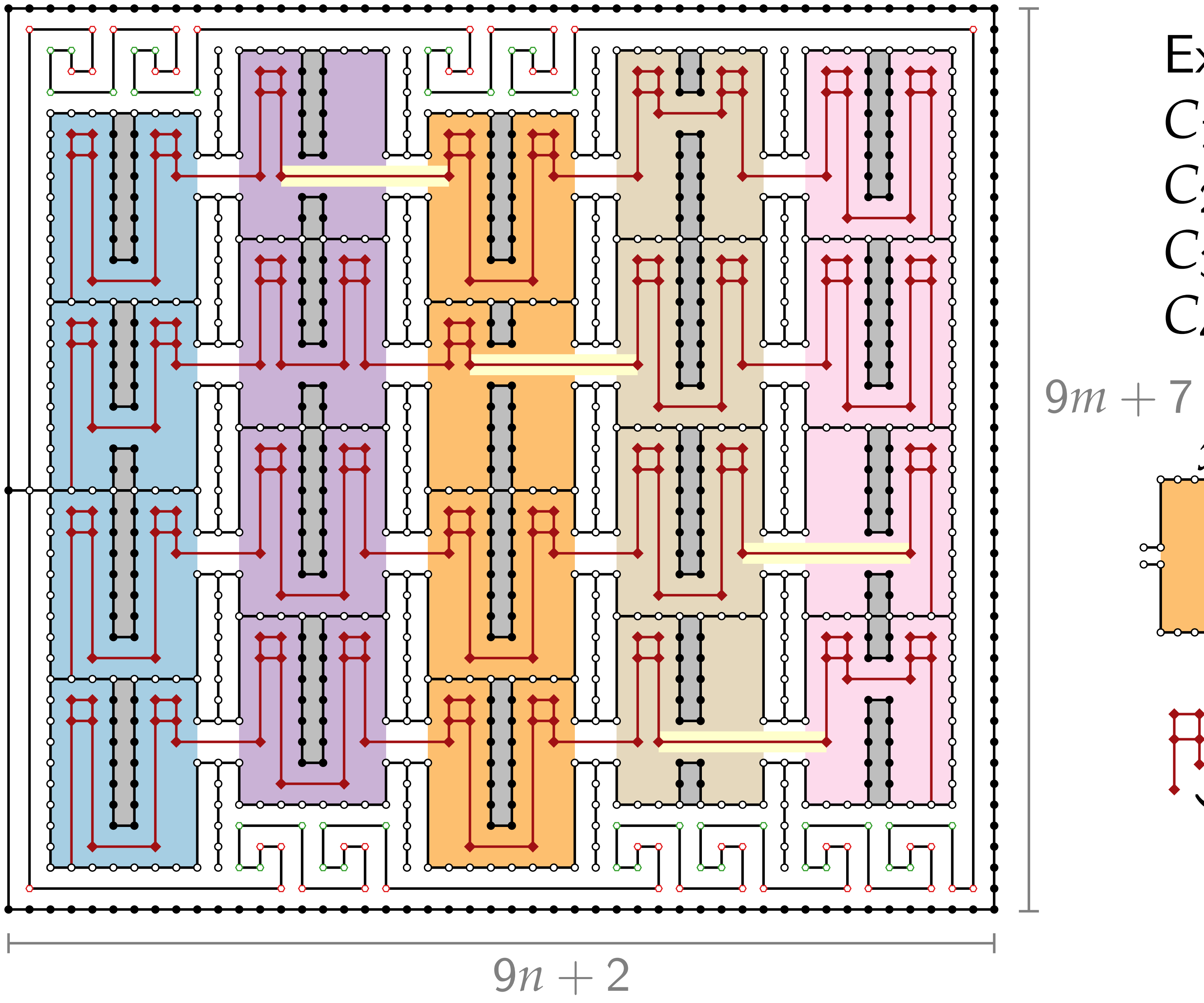
$C_3 = x_5$

$C_4 = x_4 \vee \overline{x_5}$





Full reduction



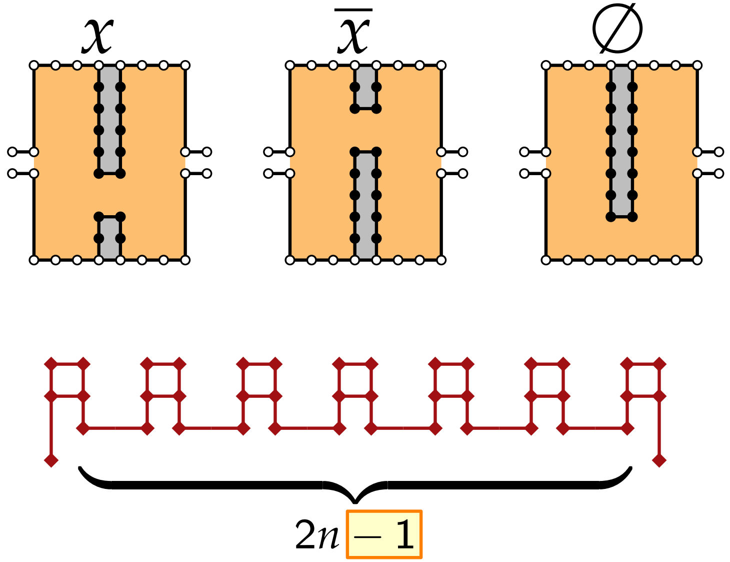
Example:

$C_1 = x_2 \vee \overline{x_4}$

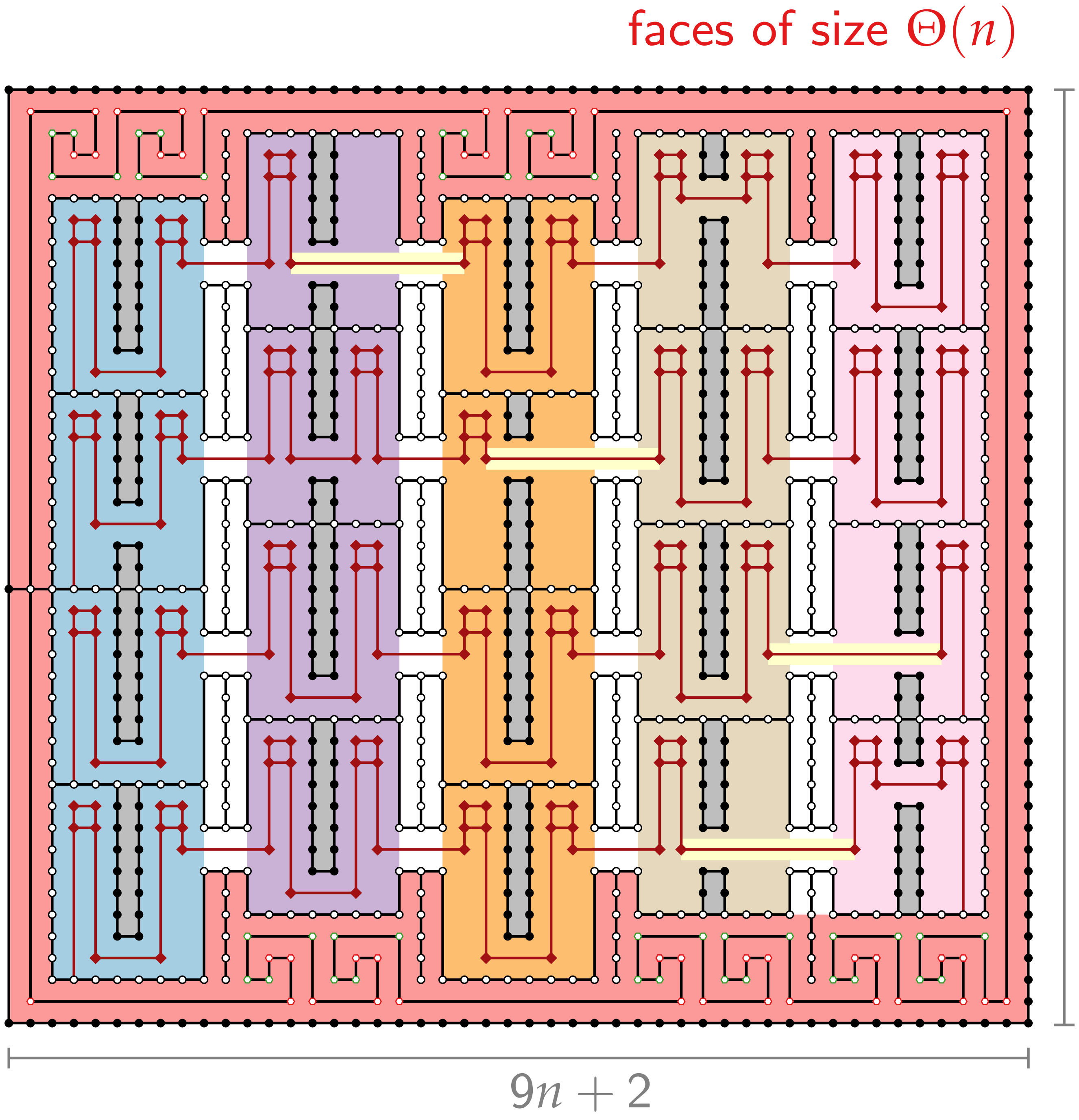
$C_2 = x_1 \vee x_2 \vee \overline{x_3}$

$C_3 = x_5$

$C_4 = x_4 \vee \overline{x_5}$



Full reduction



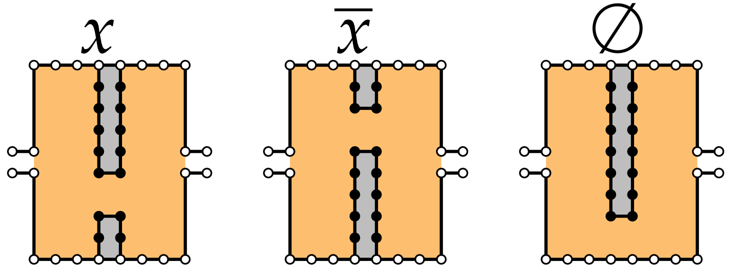
Example:

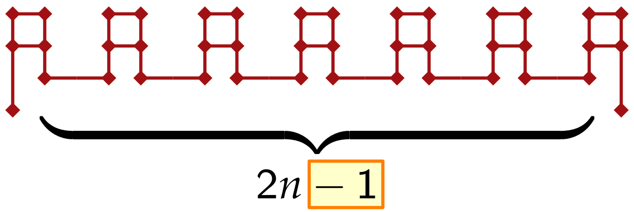
$C_1 = x_2 \lor \overline{x_4}$

$C_2 = x_1 \lor x_2 \lor \overline{x_3}$

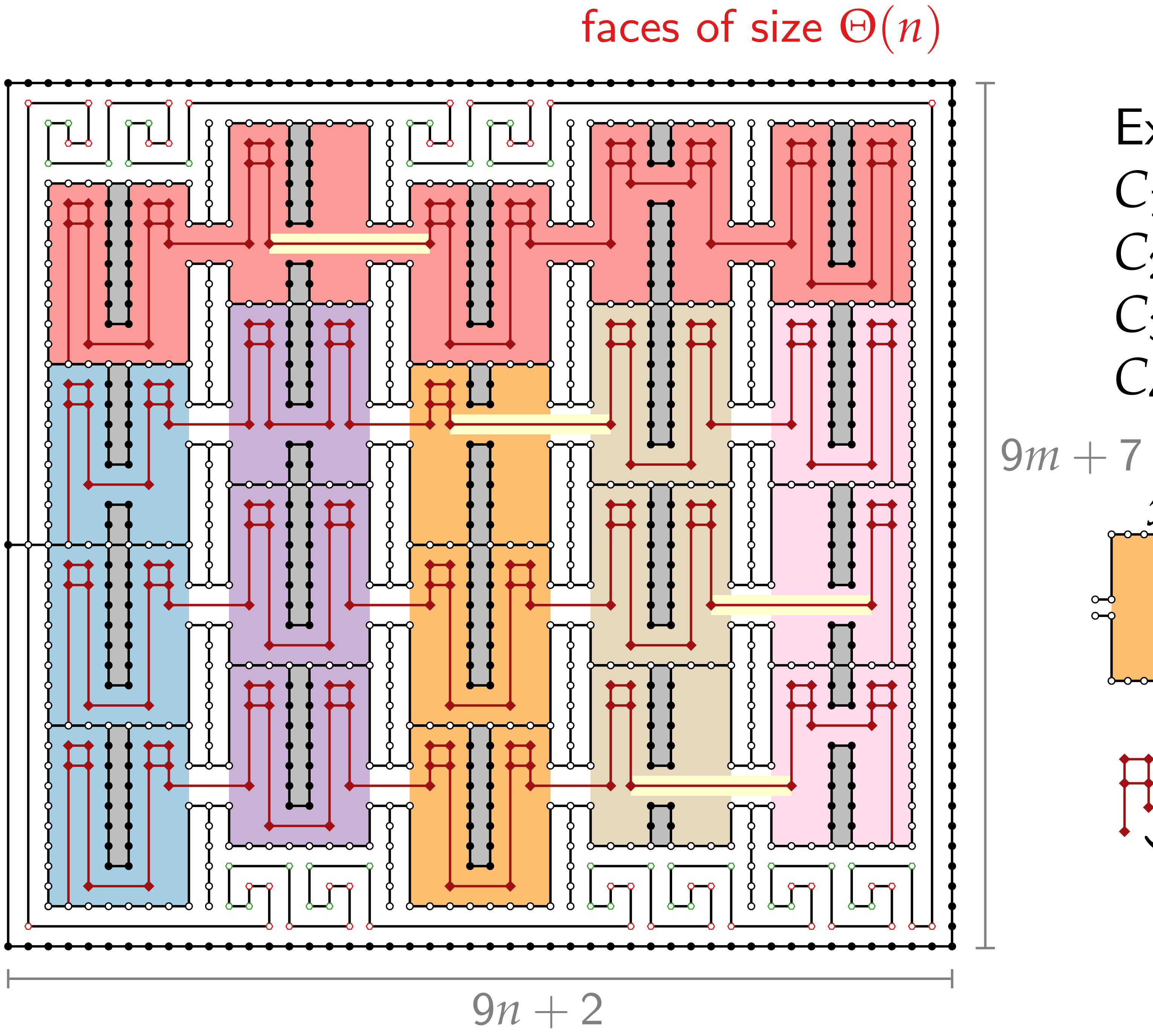
$C_3 = x_5$

$C_4 = x_4 \lor \overline{x_5}$





Full reduction



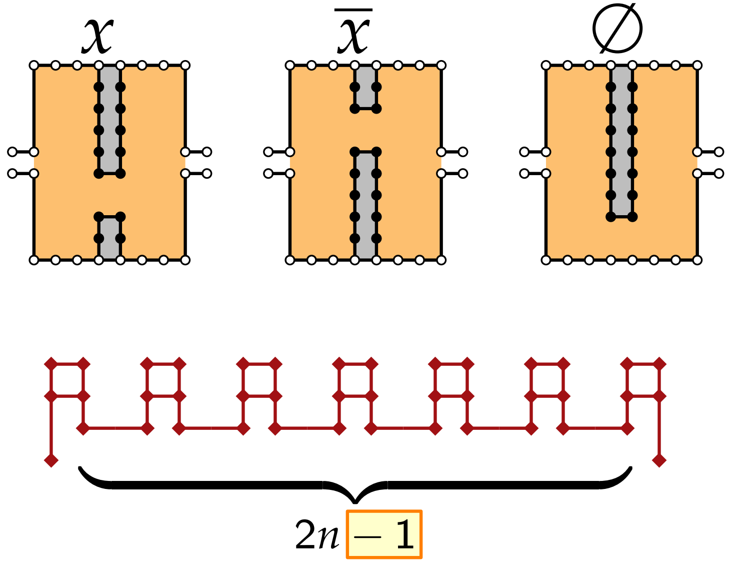
Example:

$C_1 = x_2 \vee \overline{x_4}$

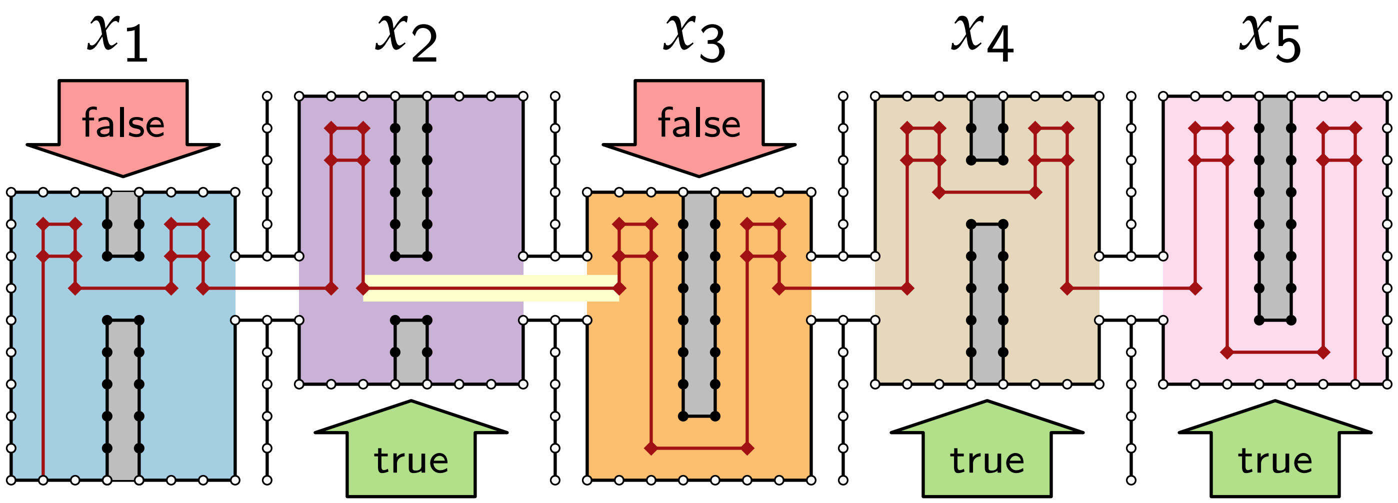
$C_2 = x_1 \vee x_2 \vee \overline{x_3}$

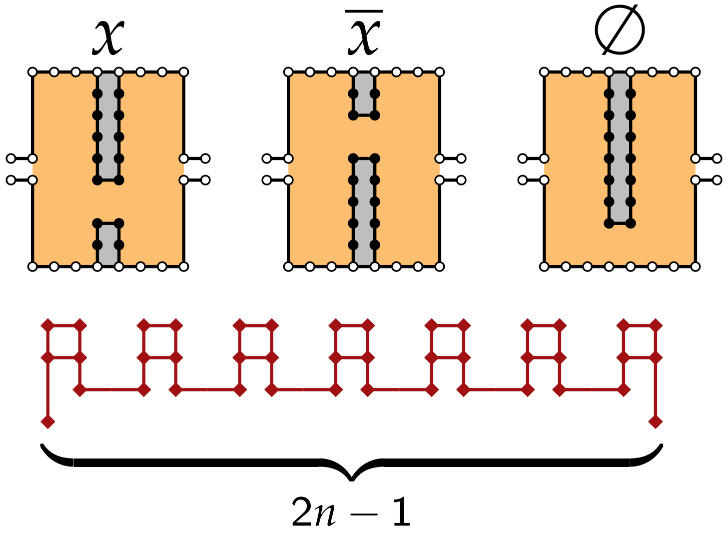
$C_3 = x_5$

$C_4 = x_4 \vee \overline{x_5}$

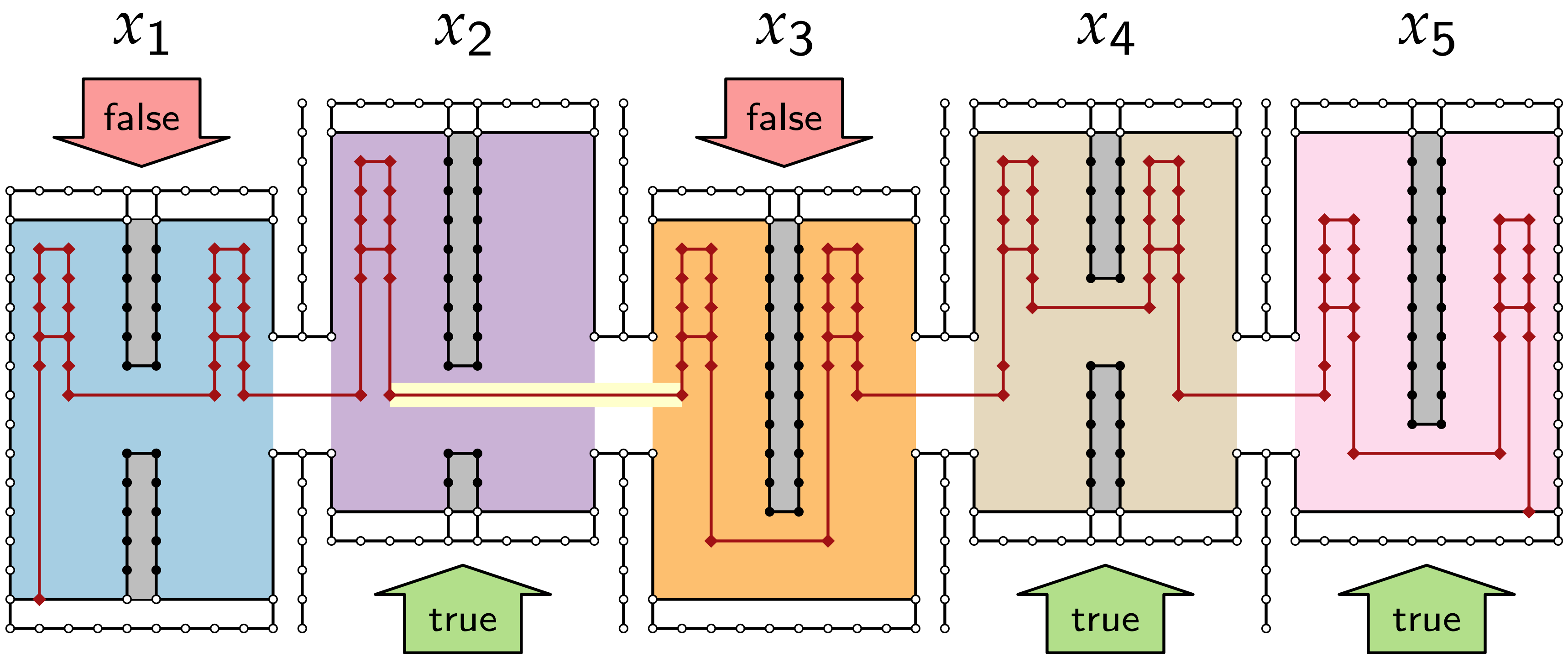


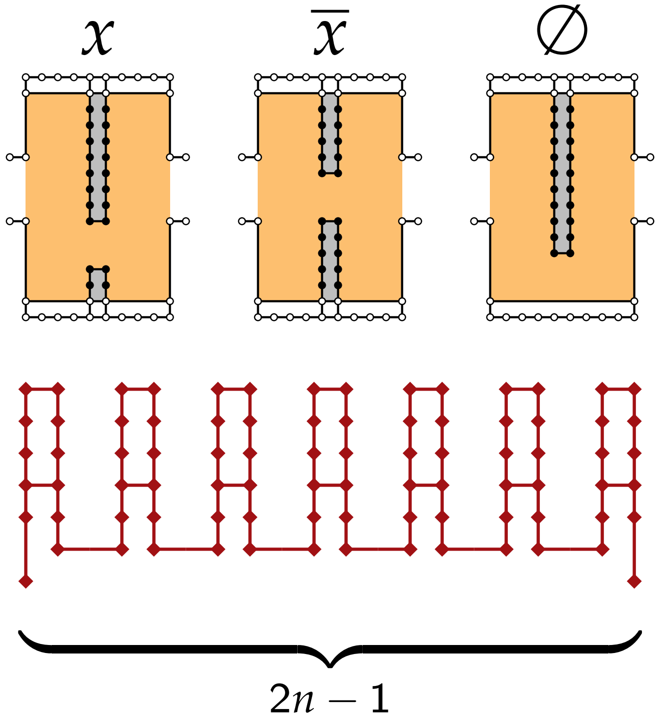
Adjusting Clause Gadgets



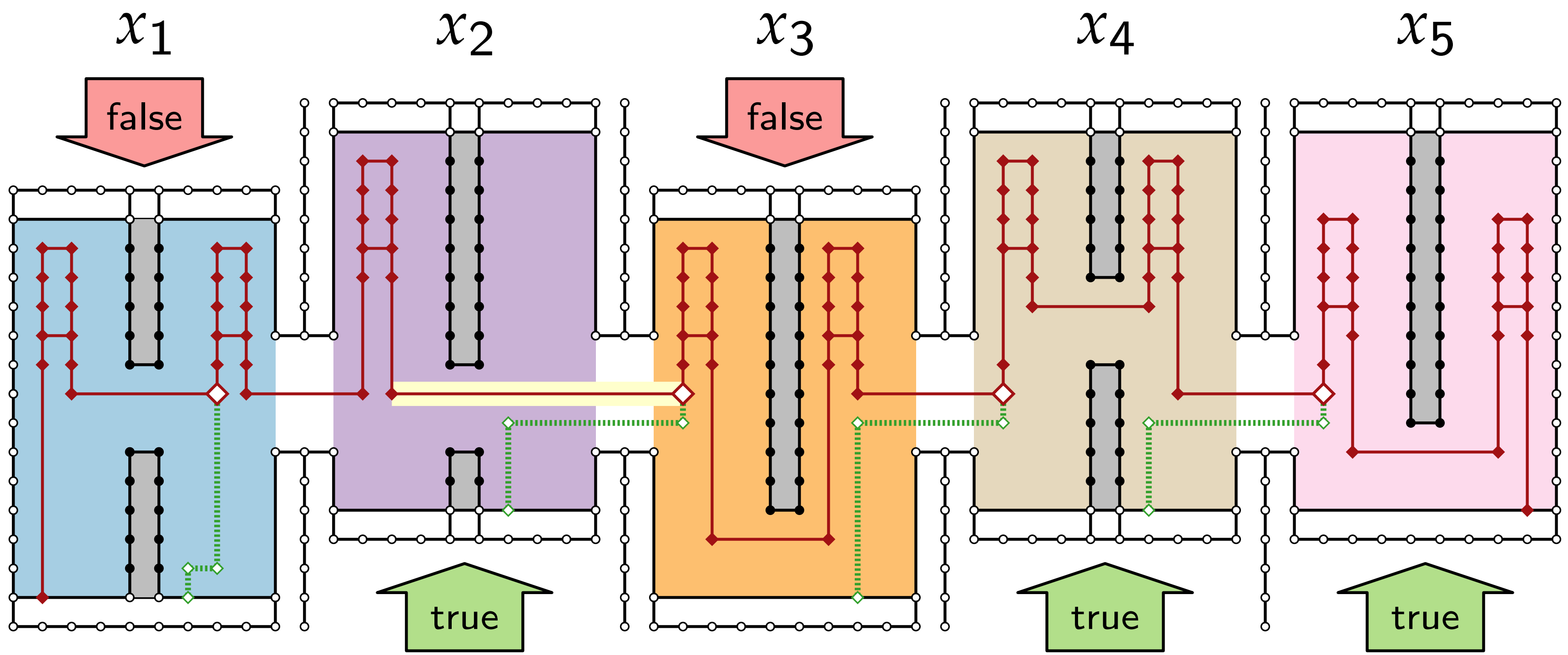


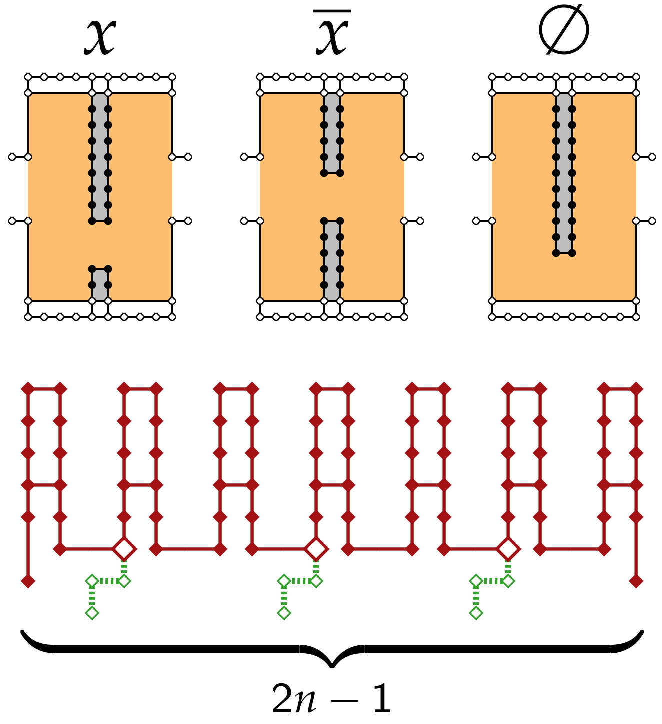
Adjusting Clause Gadgets



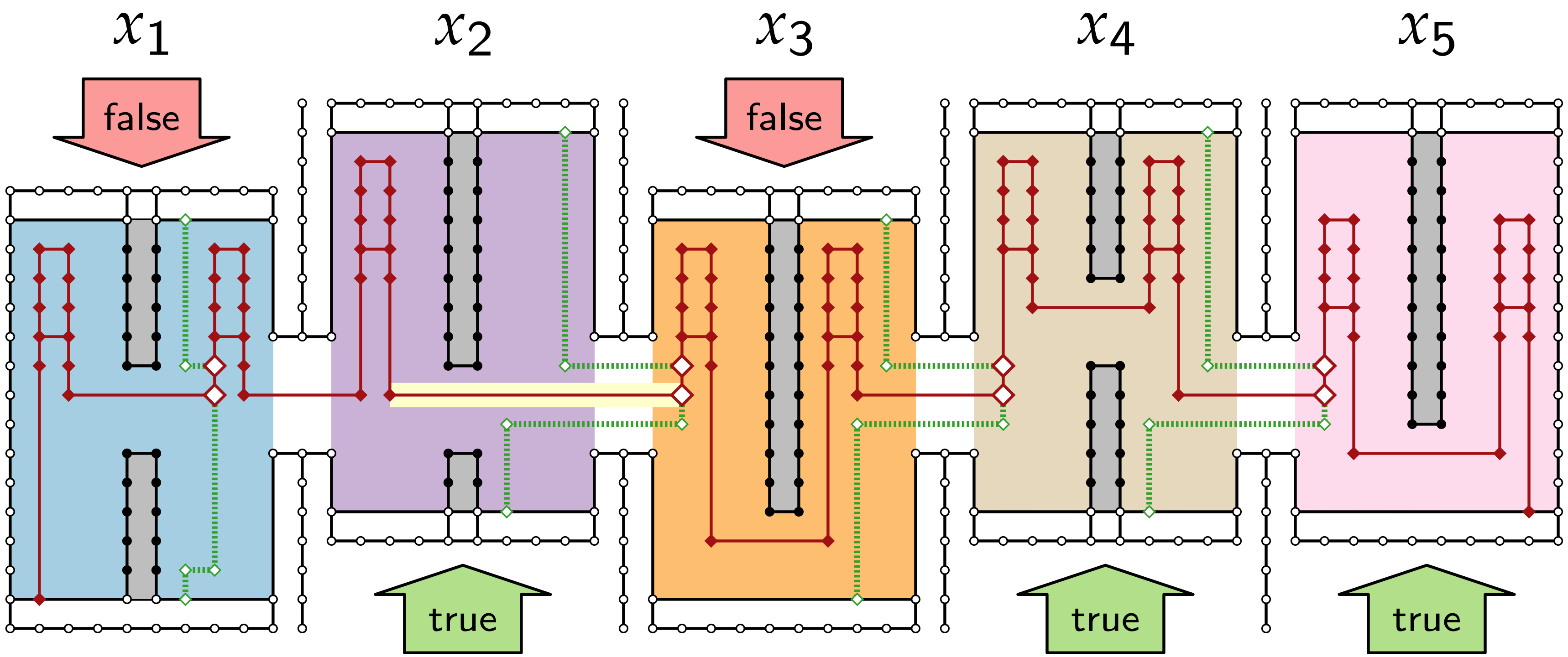


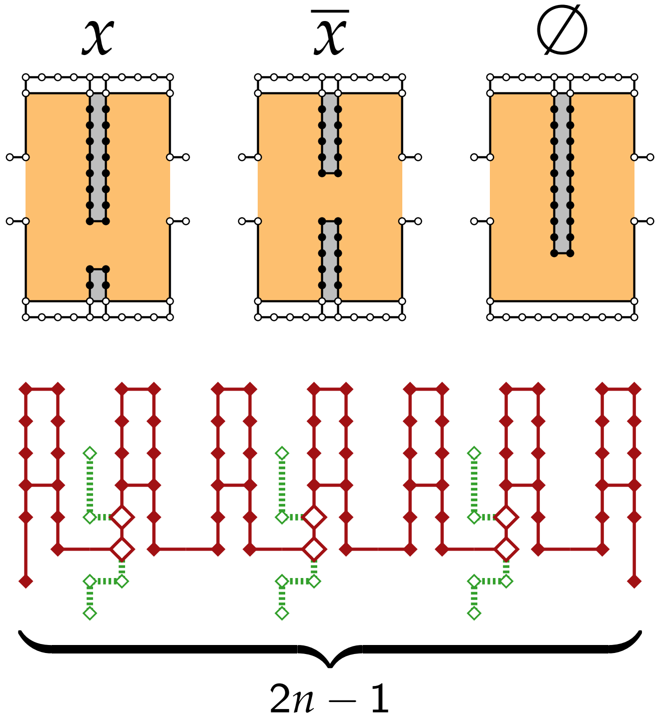
Adjusting Clause Gadgets



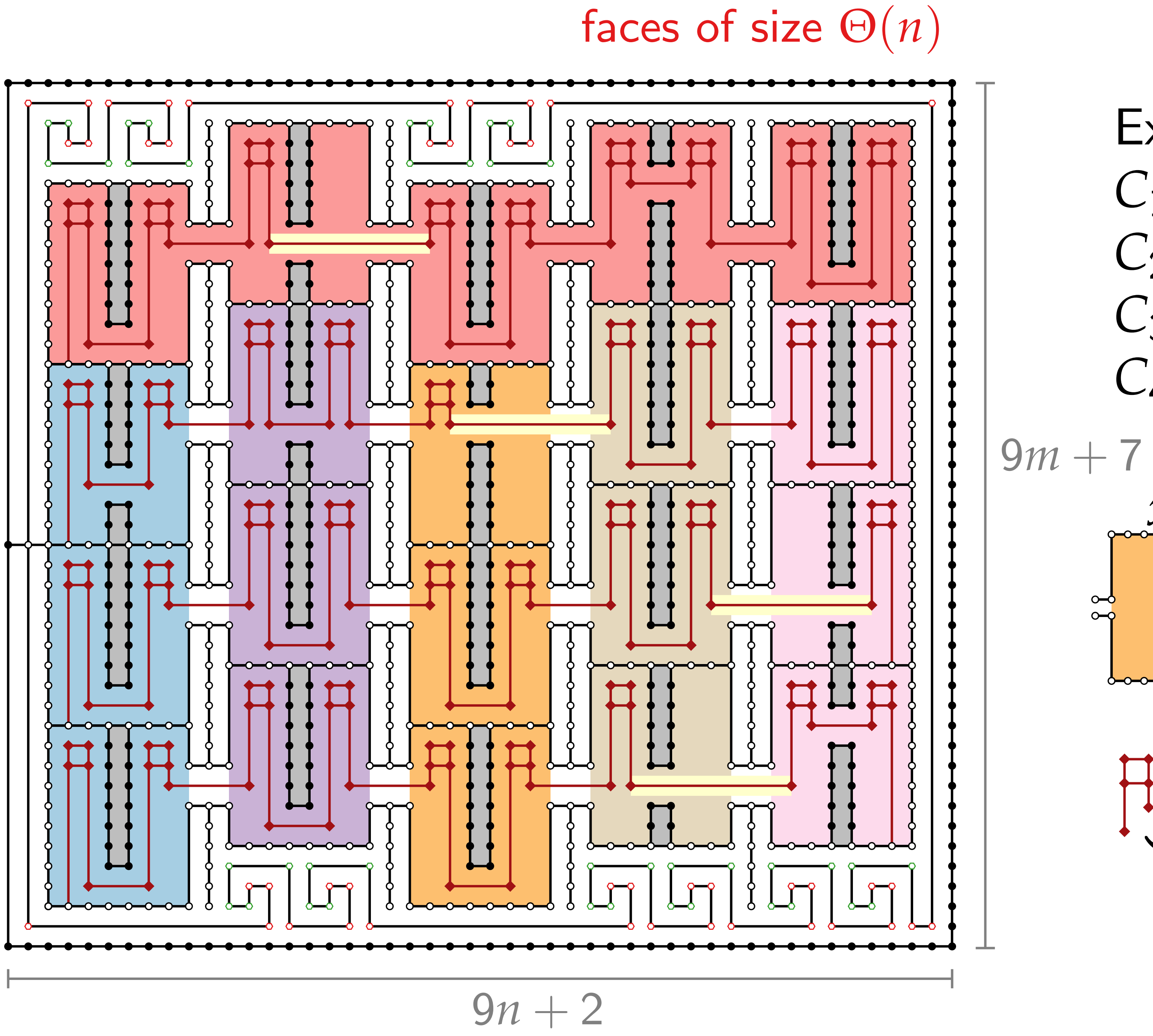


Adjusting Clause Gadgets





Full Reduction



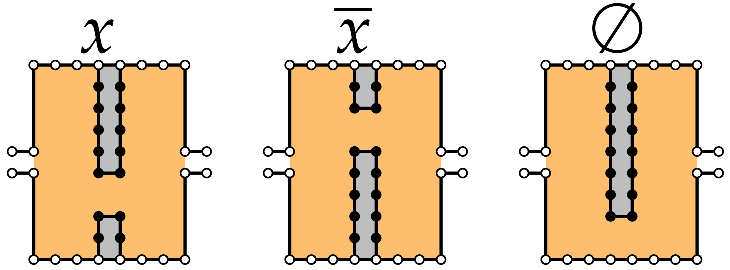
Example:

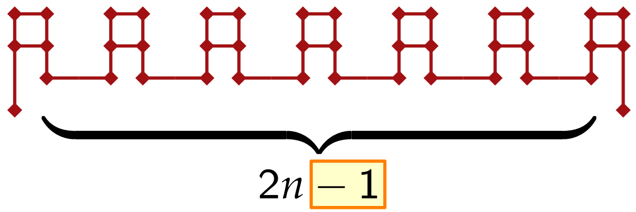
$C_1 = x_2 \vee \overline{x_4}$

$C_2 = x_1 \vee x_2 \vee \overline{x_3}$

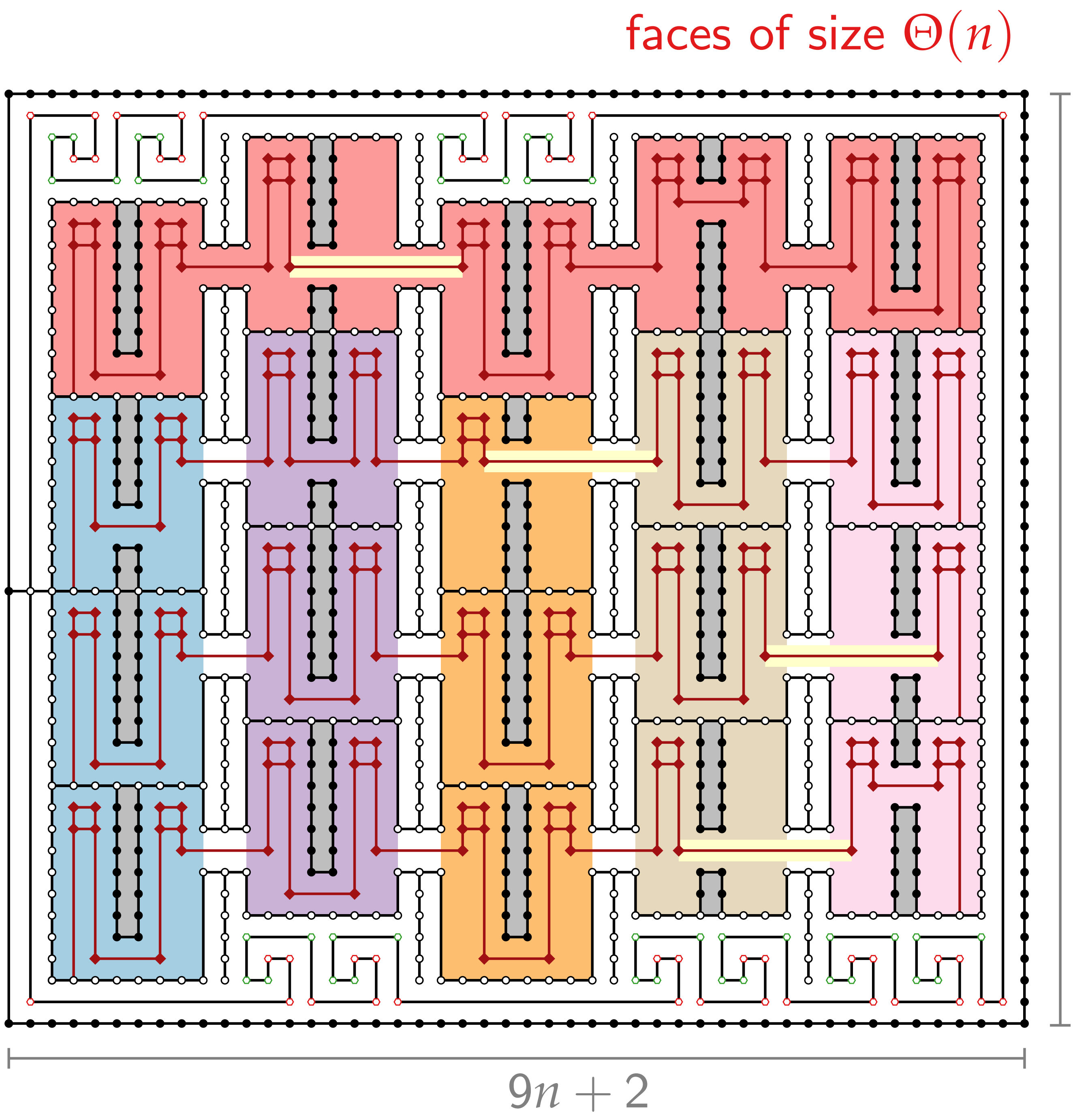
$C_3 = x_5$

$C_4 = x_4 \vee \overline{x_5}$





Full Reduction



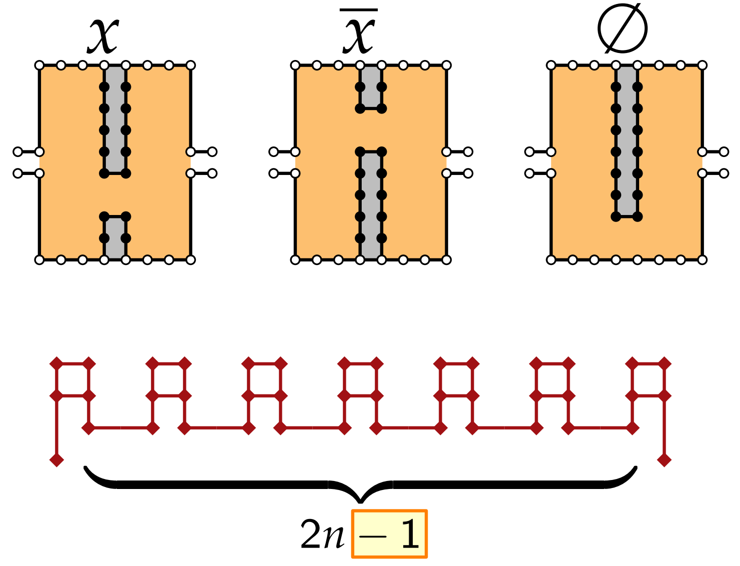
Example:

$C_1 = x_2 \vee \overline{x_4}$

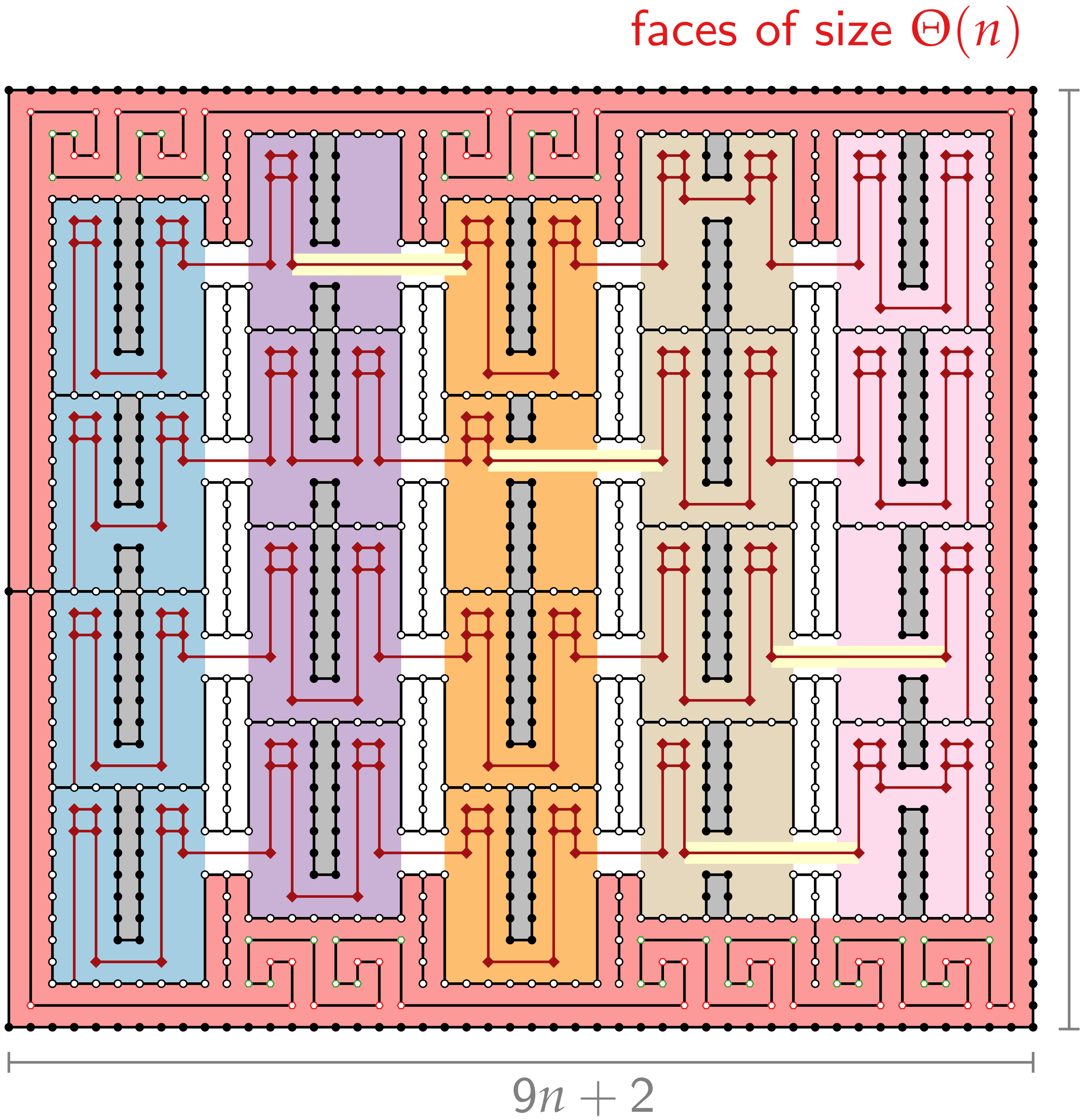
$C_2 = x_1 \vee x_2 \vee \overline{x_3}$

$C_3 = x_5$

$C_4 = x_4 \vee \overline{x_5}$



Full Reduction



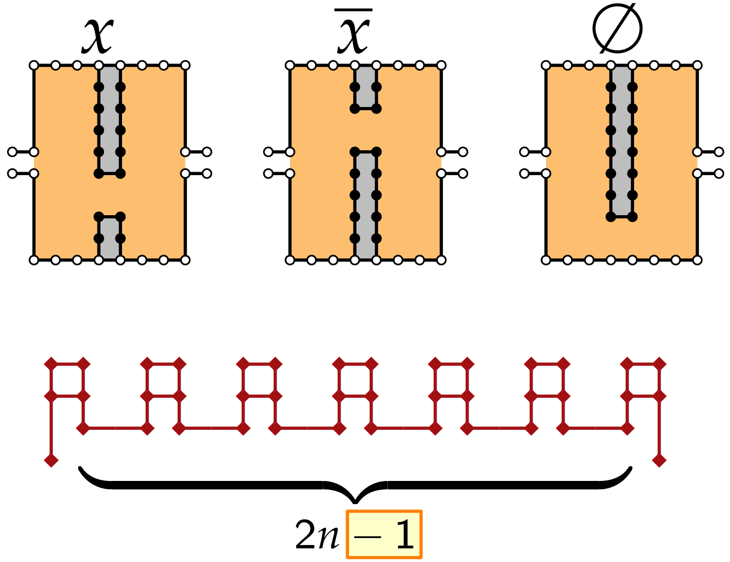
Example:

$C_1 = x_2 \vee \overline{x_4}$

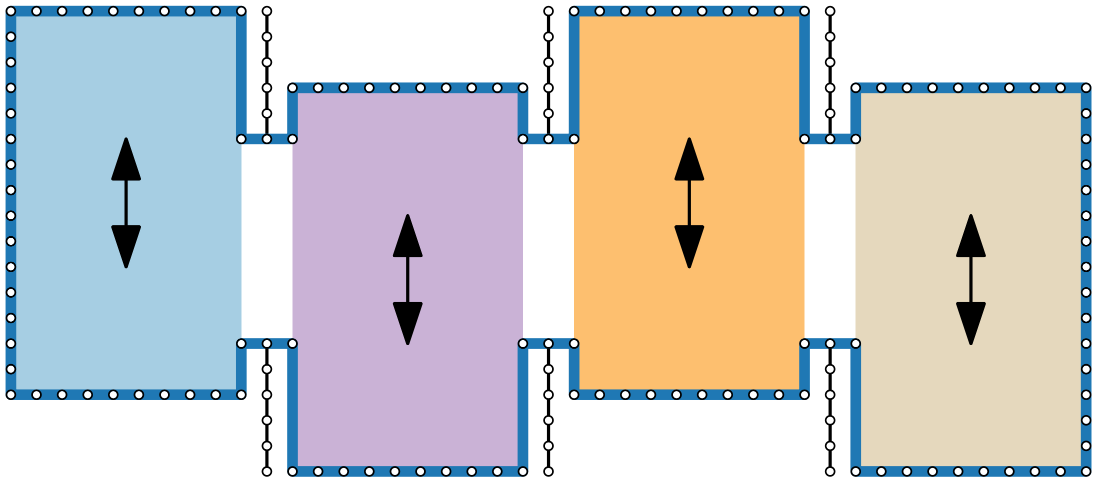
$C_2 = x_1 \vee x_2 \vee \overline{x_3}$

$C_3 = x_5$

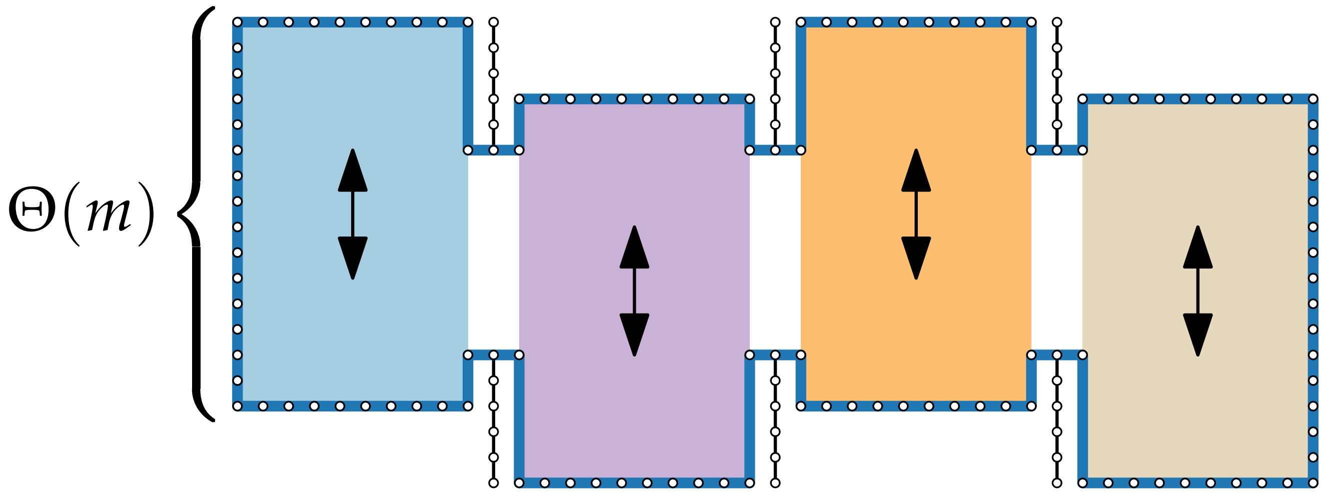
$C_4 = x_4 \vee \overline{x_5}$



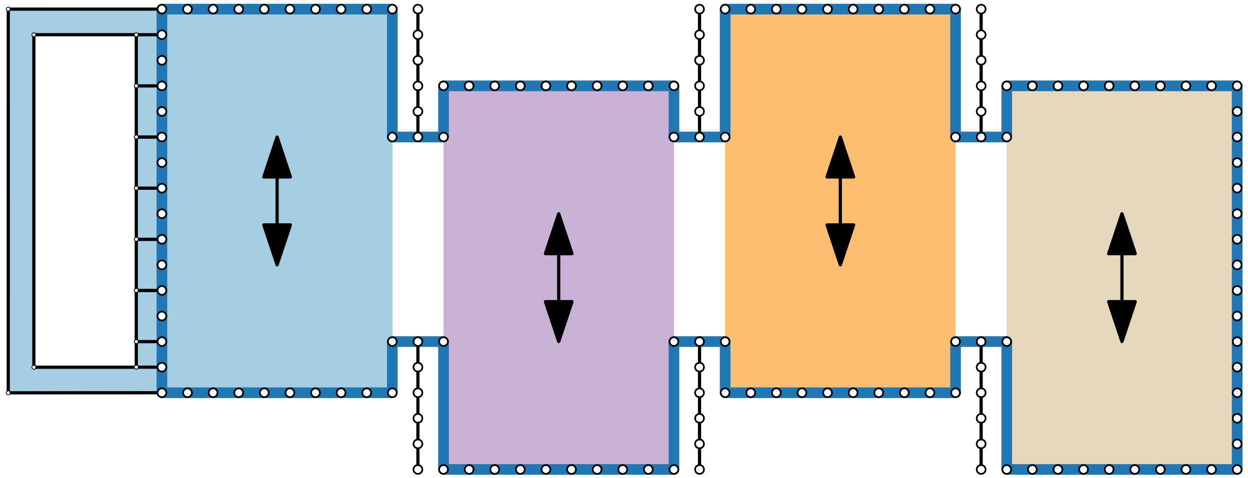
Adjusting Boundary and Belt



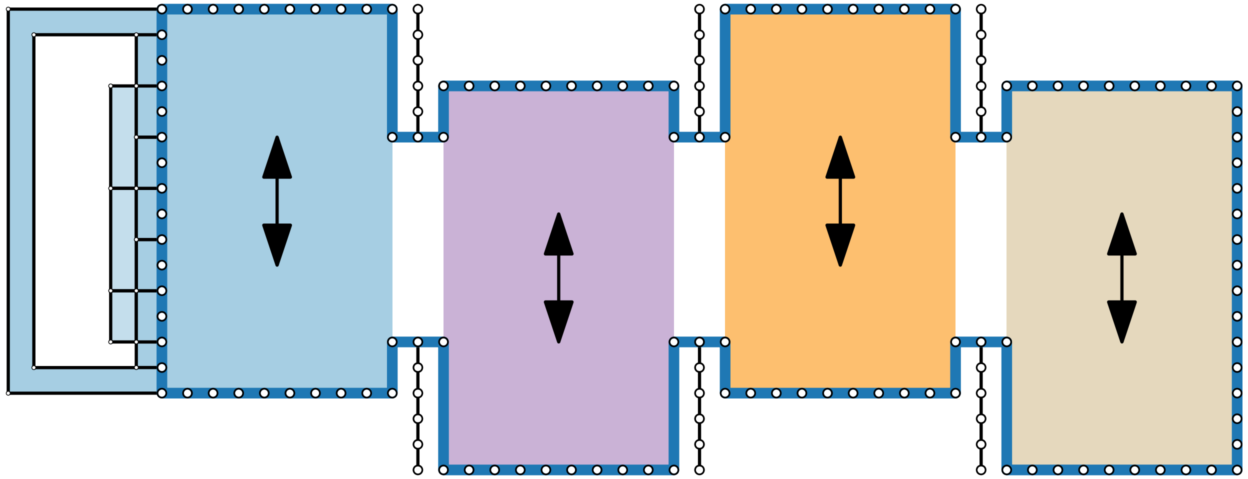
Adjusting Boundary and Belt



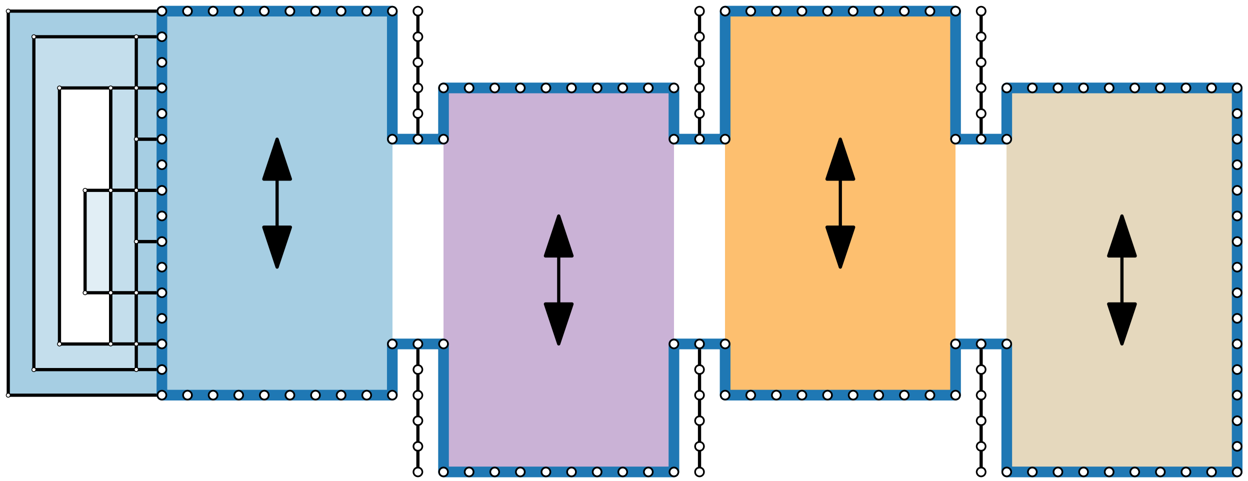
Adjusting Boundary and Belt



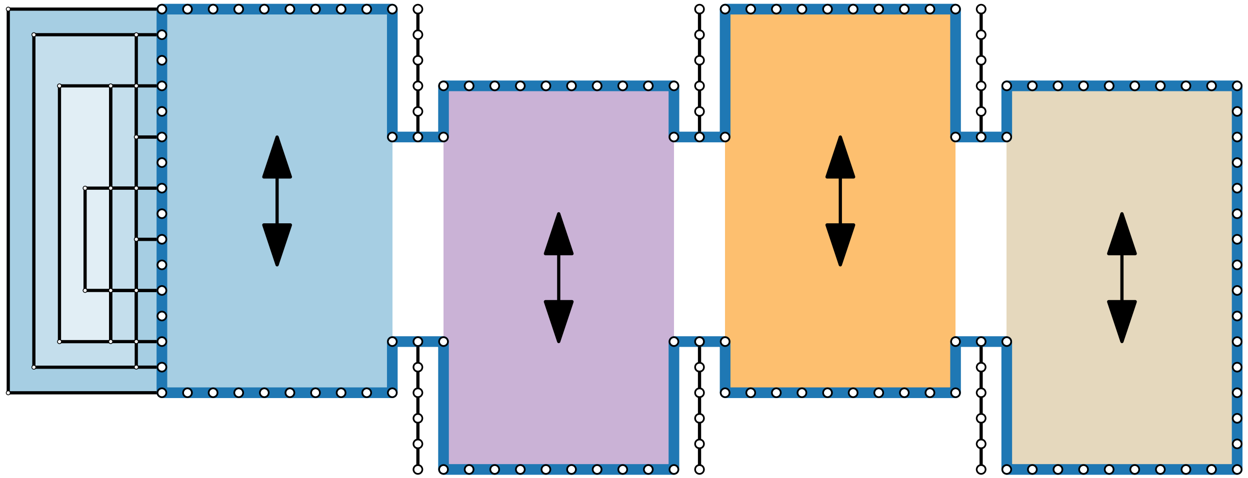
Adjusting Boundary and Belt



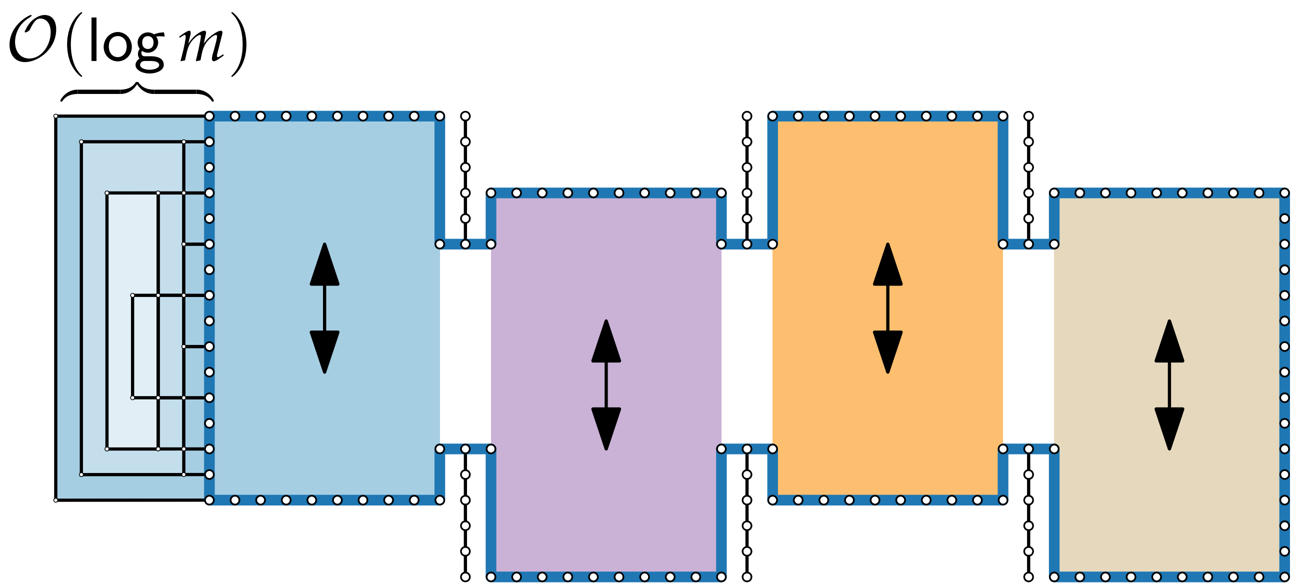
Adjusting Boundary and Belt



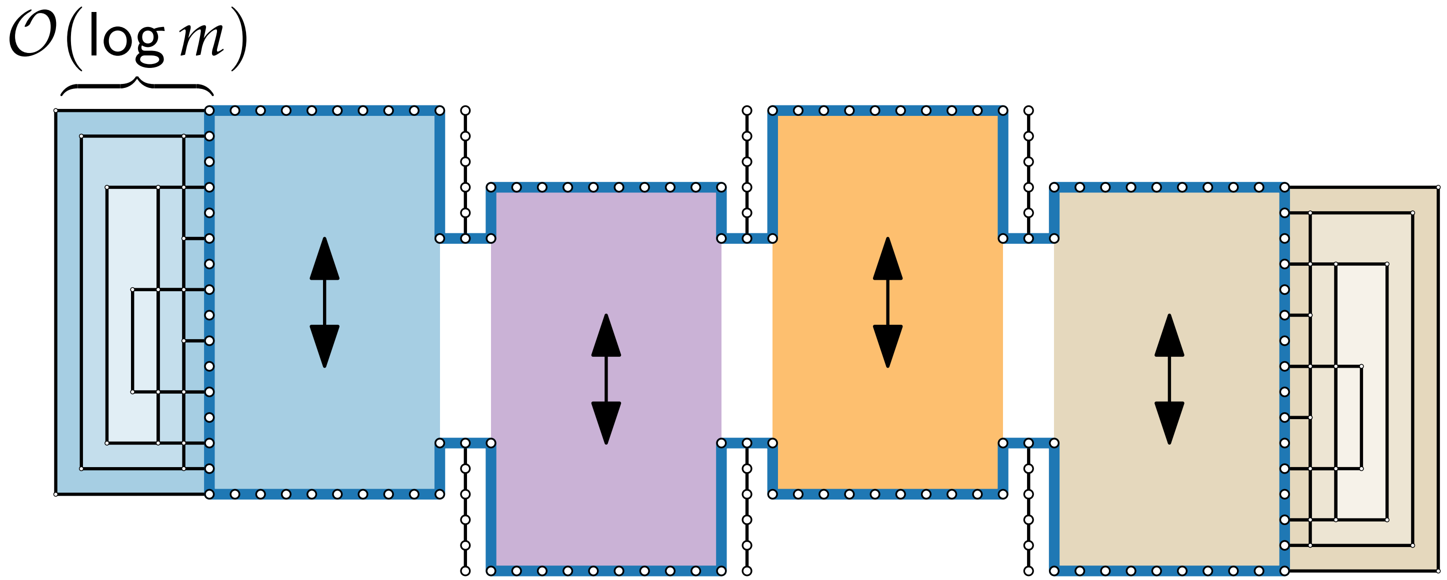
Adjusting Boundary and Belt



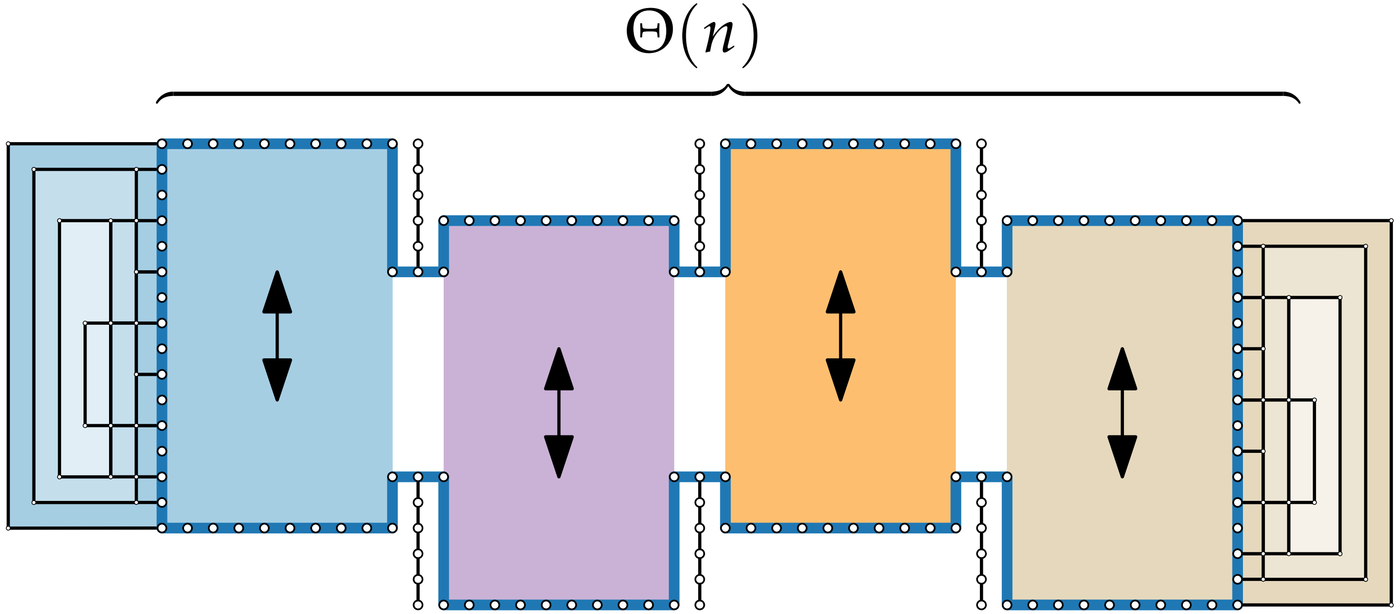
Adjusting Boundary and Belt



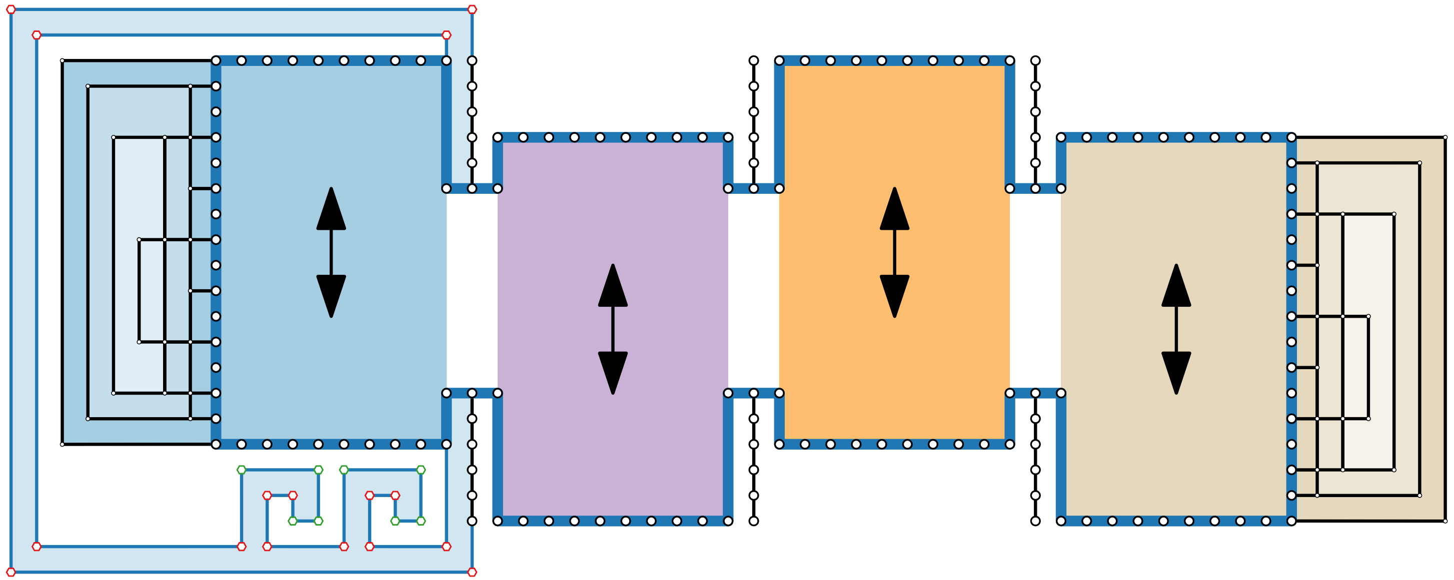
Adjusting Boundary and Belt



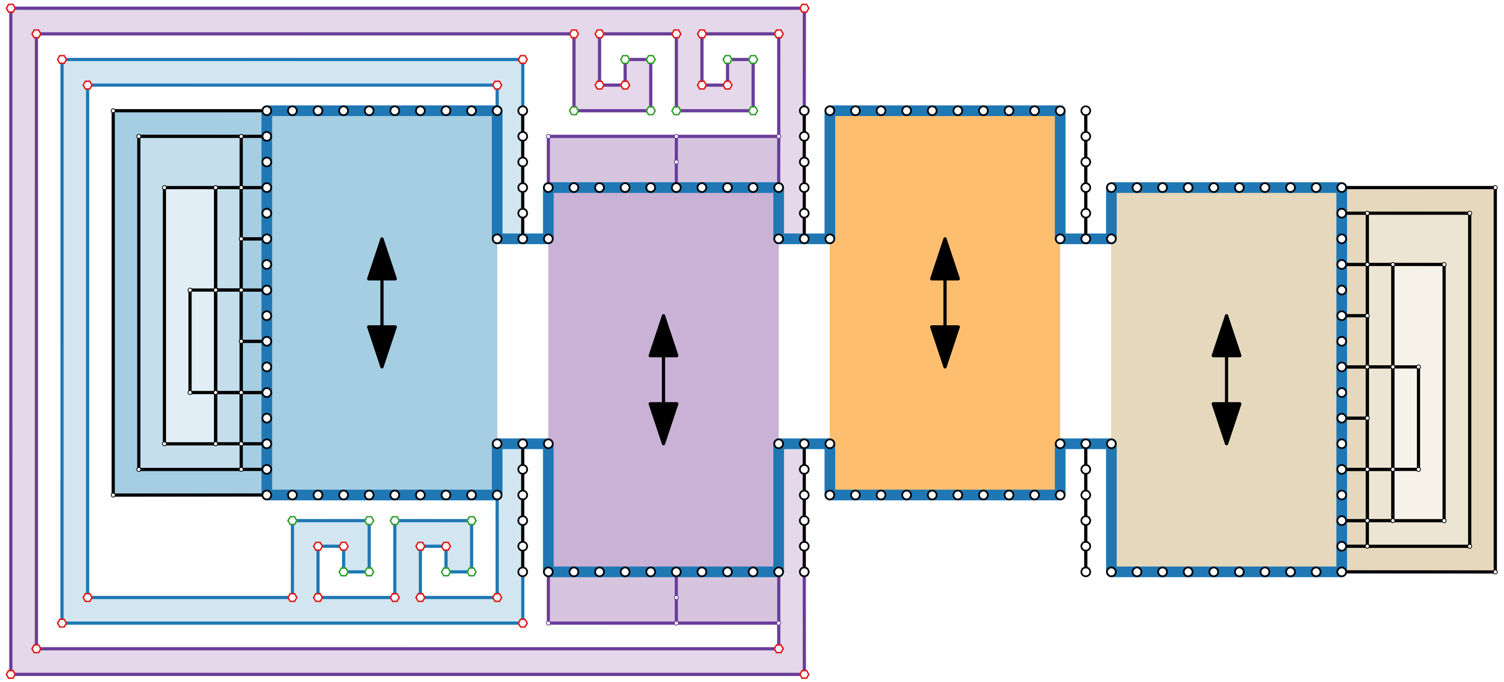
Adjusting Boundary and Belt



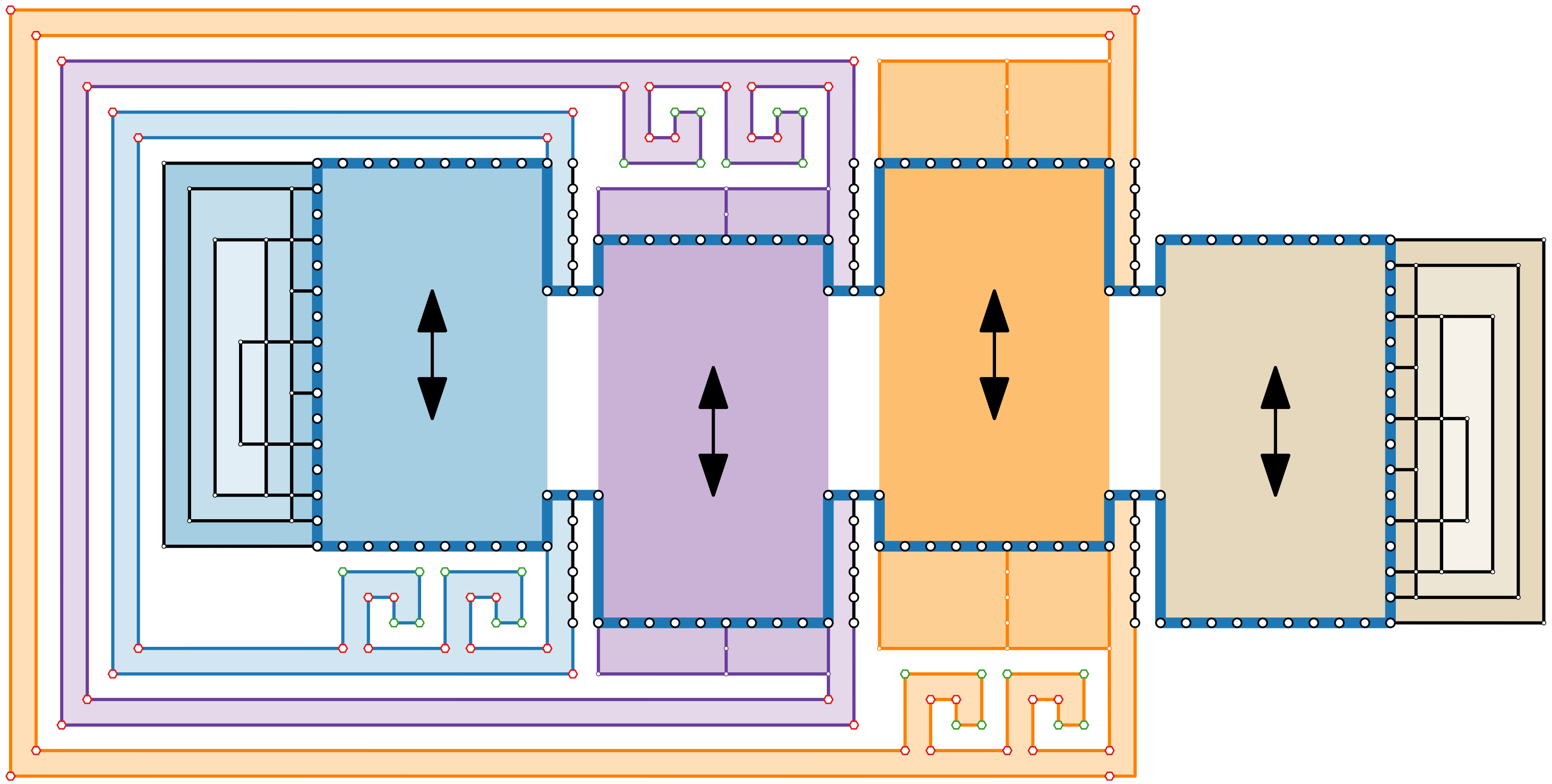
Adjusting Boundary and Belt



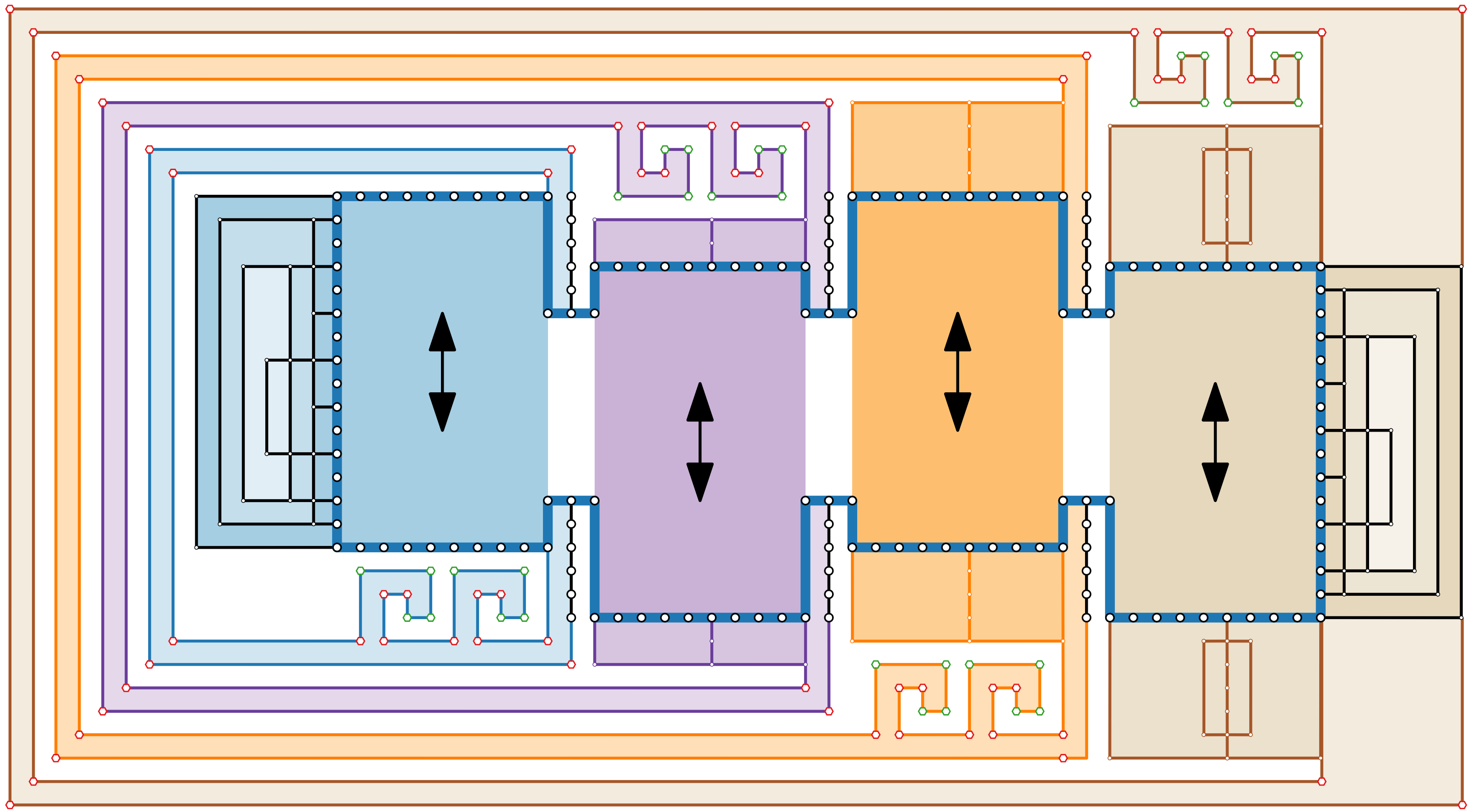
Adjusting Boundary and Belt



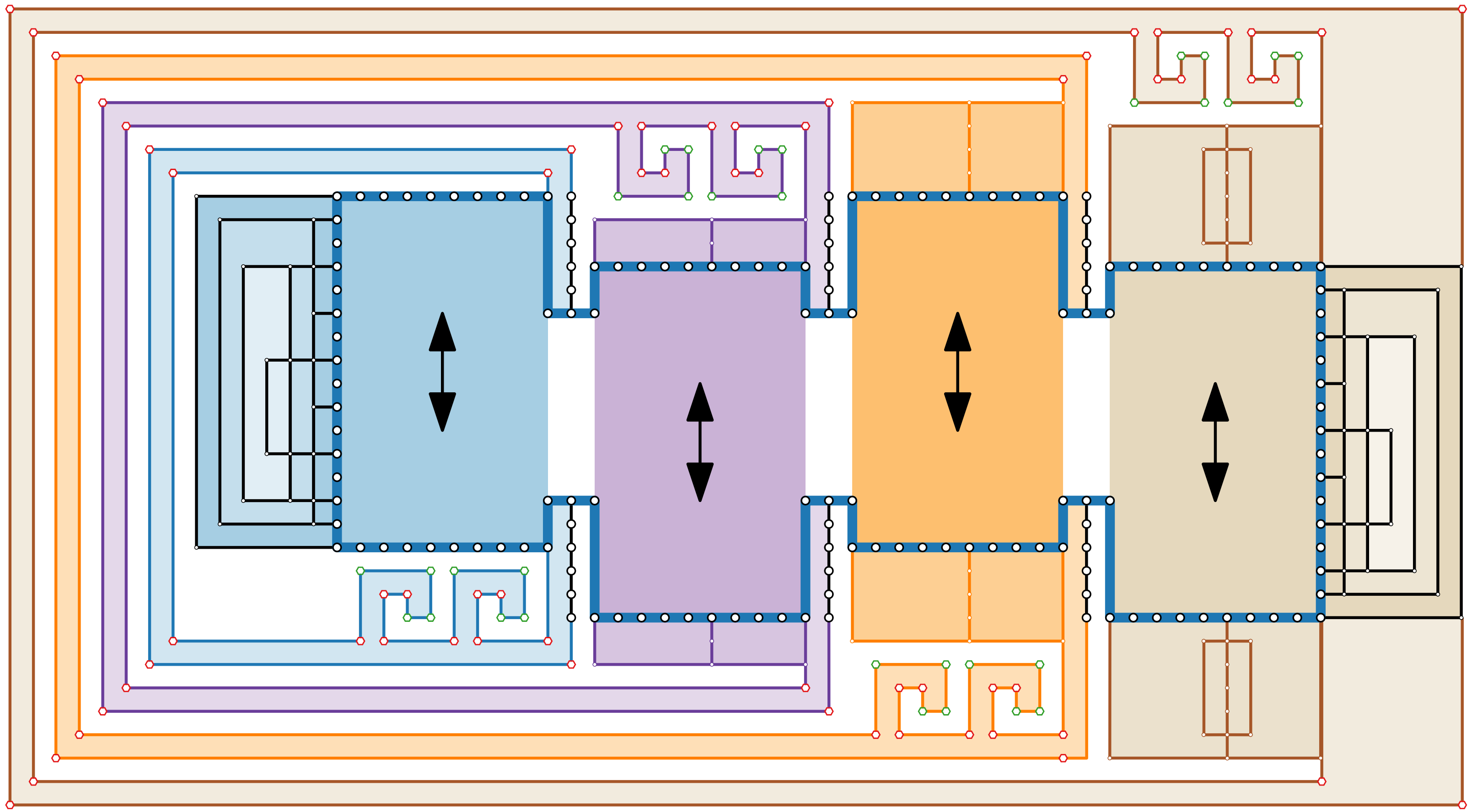
Adjusting Boundary and Belt



Adjusting Boundary and Belt

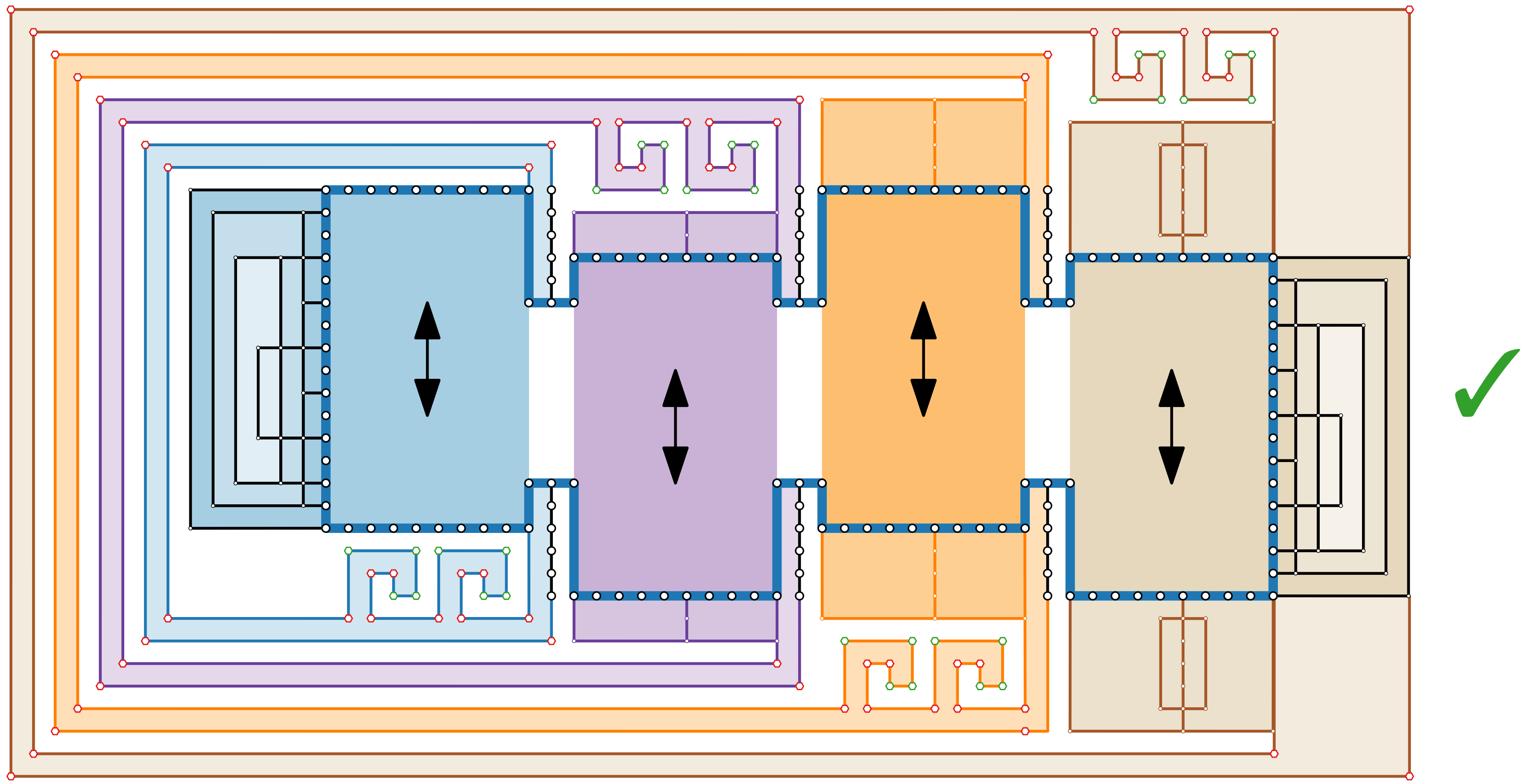


Adjusting Boundary and Belt



✓

Adjusting Boundary and Belt



Theorem. OC is para-NP-hard when parameterized by the maximum face degree.

Open Problems

- Can we find a polynomial kernel for OC w.r.t. the number of kitty corners, or at least w.r.t. the number of kitty corners *plus the number of faces*, for general graphs?

Open Problems

- Can we find a polynomial kernel for OC w.r.t. the number of kitty corners, or at least w.r.t. the number of kitty corners *plus the number of faces*, for general graphs?
- Does OC admit an FPT algorithm w.r.t. the height of the orthogonal representation?

Open Problems

- Can we find a polynomial kernel for OC w.r.t. the number of kitty corners, or at least w.r.t. the number of kitty corners *plus the number of faces*, for general graphs?

- Does OC admit an FPT algorithm w.r.t. the height of the orthogonal representation?

- Is OC solvable in $2^{O(\sqrt{n})}$ time?
  This bound would be tight assuming that the Exponential Time Hypothesis is true.

Open Problems

- Can we find a polynomial kernel for OC w.r.t. the number of kitty corners, or at least w.r.t. the number of kitty corners *plus the number of faces*, for general graphs?

- Does OC admit an FPT algorithm w.r.t. the height of the orthogonal representation?

- Is OC solvable in $2^{O(\sqrt{n})}$ time?
  This bound would be tight assuming that the Exponential Time Hypothesis is true.

- If we parametrize by the number of *pairs* of kitty corners, can we achieve substantially better running times?